MURDOCH
UNIVERSITY
PERTH, WESTERN AUSTRALIA

Faculty of Minerals and Energy
School of Engineering and Energy

ORE CAR LASER SCANNER

Engineering Internship Project

Graham West

November 2008

Declaration

Except where I have indicated, the work I am submitting in this assignment is my own and has not been submitted for assessment in another course.

Abstract

This report discusses the development of a system using laser scanning technology to examine railway rolling stock. The scanner is to be used to examine ore cars after having been emptied. Principal motivation for this project is to identify ore cars containing residual ore, after being emptied or damaged cars, in order to prevent derailments. The system is based around a SICK laser scanner mounted above the rails which scans the cars as they pass underneath. Data obtained from the scan is processed by an embedded PC. This report examines laser scanners in general, discusses the calculations made by the embedded PC and discusses the implementation into an existing site. The report also addresses the problematic components of the system including a suitable search tool to identify the sides of the car and communication interfaces between the various pieces of equipment.

Acknowledgements

The author would like to thank the following people for their assistance and support in bringing this project and report to fruition:

Mr Bruce Johnson: BEC Engineering

Mr Simon Dreyer: BEC Engineering

Mr Adam McConkey: BEC Engineering

Professor Parisa Bahri: Murdoch University

Associate Professor Graeme Cole: Murdoch University

Doctor Gareth Lee: Murdoch University

Audrey West

Table of Contents

Declaration....1
Abstract....2
Acknowledgements....3
Table of Contents....4
1.0 Introduction....6
1.1 Background....6
1.2 Objective....7
1.3 Scope of work....8
1.4 Concept....9
1.5 System development and testing....10
1.6 Company Profile....12
1.7 Project Team Members....13
2.0 Laser Scanner....15
2.1 Theory of operation....15
2.2 Data Communications....16
2.3 Baud Rates....17
2.4 Retrieving scan values....19
2.5 Send telegrams....20
2.6 Receive Telegrams....21
2.7 Turning the laser on and off....22
3.0 Embedded PC....22
3.1 Hardware and operating system....22
3.2 Processing the results....23
3.3 Triggering....26
3.4 Finding the profile....26
3.5 Profile Integration....31
3.6 Alignment....33
3.6.1 Skew....33
3.6.2 Level....34
3.7 Database....35
3.7.1 Database Software Functions....37
3.7.2 View database mode....38
3.8 Alarm Loop....39
3.9 PLC Communications....39
3.10 Error Handling....40
4.0 SCADA....41
4.1 SCADA Communications....41
4.2 Charting....42
4.3 Buffering scan data....45
4.4 Viewing historical results....48

5.0 Project Management 49
5.1 Progress 49
5.2 Scope of work changes 50
5.3 Future Work 51
6.0 Summary 52
References 54
Appendixes 56
Appendix i – Notes to the accompanying CD ROM. 56
Appendix ii – Procedure for changing scanner baud rate 57
Appendix iii – Sample of raw test data for finding the side 58
Appendix iv – embedded PC calculation flow chart 59
Appendix v – sample of raw test data, finding the side(2) 60
Appendix vi – Sample of database results table 62
Appendix vii – Alarm flow chart 63
Appendix ix – Citect tag list 64
Appendix x – sample of raw data – test Citect communications 66
Appendix xi – Site acceptance test procedure 67

1.0 Introduction

The aim of this report is to discuss the development of a system using laser scanning technology. The motivation for this project is to reduce the occurrences of train derailments caused by unbalanced loads in open rolling stock. A SICK laser scanner is to be used to examine the rolling stock after having been emptied. Data from the laser scanner will be interpreted by an embedded PC which is interfaced into existing control system infrastructure. The development of this concept has been made using models of rolling stock in a Perth workshop by BEC engineering.

1.1 Background

In some mining operations ore is transported by rail. The ore is transported by open rolling stock from a mining operation to a processing plant or port for export. This project focuses on the unloading operation of these railway ore cars. The cars are unloaded by rotating the entire car allowing the contents to fall into a hopper. The ore is then transported to a stockpile by conveyor belt. During this operation cars are disconnected from the train, pushed into the cell and then rotated upside down. Once the contents have fallen out the cars are returned to the upright position and ejected from the cell. Figure 1.1 shows an example of an unloading cell. This operation is fully automated using a PLC but monitored from an operations room via a CCTV network and a SCADA network.

Figure 1.1 - Ore car unloading cell.

When ore becomes damp it turns to mud and can cling to the inside of the car. This ore does not always completely fall out, when the car is inverted. During normal operation if excessive amounts of residual ore are still present in the car after unloading, the car can be returned to the tipping cell and re-emptied. A decision to re-tip an ore car lies at the discretion of operation staff who are watching the unloading process via CCTV (closed circuit television). The re-tip operation is invoked manually by staff if necessary. (BEC 2008a)

Failure to detect cars with excessive amounts of residual ore has resulted in several serious train derailments. Investigations have found that as the ore car was being returned to the upright position the residual ore became dislodged and fell to one side of the car. These cars have a tare of 18 tonnes which can hold up to 130 tonnes of ore. When the ore has been piled up on one side of the car the shift in centre of gravity can be sufficient to cause the car to become derailed. (BEC 2008a)

Train derailments have a serious economic impact on business operations. Apart from the damage to railway infrastructure and rolling stock, derailments also cause loss to production and can pose series threat to staff safety. This project discusses the development of a system to determine if ore has been left in a car after it has been emptied. This system will help to prevent train derailments leading to a safer working environment and greater production levels. (BEC 2008a)

1.2 Objective

The two main objectives of this project are staff safety and economic loss prevention. The proposed system uses a laser scanner to profile the cars as they are ejected from the tipping cell. This equipment will be used in the early detection of a situation which could lead to a train derailment. In the event of excess residual ore or where the ore is piled on one side of the ore car, an alarm event will be generated to alert operations staff of the situation. Operations staff will then have a numerical measurement for the amount of residual ore and a 3 dimensional chart showing the profile of the ore car contents. This will enable staff to make an informed decision on whether to re-tip an ore car. Presently the use of CCTV is difficult during inclement weather and the evening shifts (BEC 2008b).

An objective for the longer term is to be able to use this numerical information as part of the control system. In the event that the amount of residual ore or degree of imbalance exceeds certain set points, the control system could return the car to the tipping cell to empty any residual ore. A return to the tipping cell needs to be conditional to the ore car still being on the rails. If an ore car is already derailed when it is returned to the tipping cell damage to the tipping cell will result (BEC 2008b).

This project entails development of a system enabling the detection of:

1) The quantity of residual ore in the car after the car has left the tipper.
2) The extent of which any residual ore would be unevenly distributed.
3) If the ore car is on the rails and undamaged after it has left the tipping cell (BEC 2008b).

1.3 Scope of work

The principal instrument used to examine the ore cars will be a laser scanner produced by SICK technologies (SICK 2006b) in Germany. The scanner will be interfaced to an embedded PC which interprets the results and records these into a database. In addition, part of this project is to interface this system to the existing PLC on site, which controls the car dumper. The existing on site SCADA system used by operations staff to monitor operations will also need to be reprogrammed to accommodate the laser scanning system (BEC 2008 b). An interface will also need to be developed for communications between the SCADA system and the embedded PC. A block diagram of the components for this system is shown in figure 1.2.

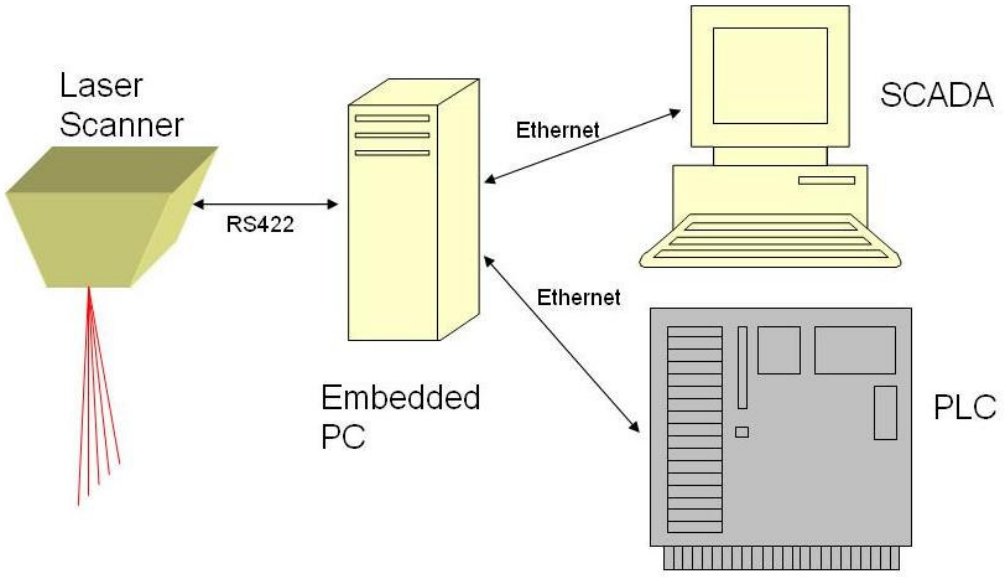


Figure 1.2 – Hardware and communications block diagram.

This report describes the development and interconnection of the various components which had to be researched, procured, adapted or designed, to generate an overall working solution. Specifically this includes:

1) Interfacing the laser scanner with the embedded PC
2) Interfacing the existing PLC with the embedded PC
3) Interfacing the existing Citect SCADA with the embedded PC
4) Development of software for the embedded PC
5) Building a Database on the embedded PC
6) Modification to the existing Citect software

The scope of this report covers the research and development of a stand alone turnkey product. While this system has been developed for a specific client, this report is of a general nature intended only to cover the research and development of a product to deal with a common problem in the mining industry. At the time this report was compiled the system had only been tested in a workshop environment and had not been tested on a working mining operation.

1.4 Concept

A laser scanner is a non contact distance measurement device which will examine the ore cars (SICK 2006b). The scanner measures distances to multiple points through a radial arc. The scanner is mounted above the ore cars so that the arc is at right angles to the car as shown in figure 1.3. The cross-sectional profile is scanned several times as the ore car passes underneath. The scanner output is a set of polar co-ordinates which are converted to Cartesian co-ordinates and then integrated to determine the cross sectional area of the ore car.

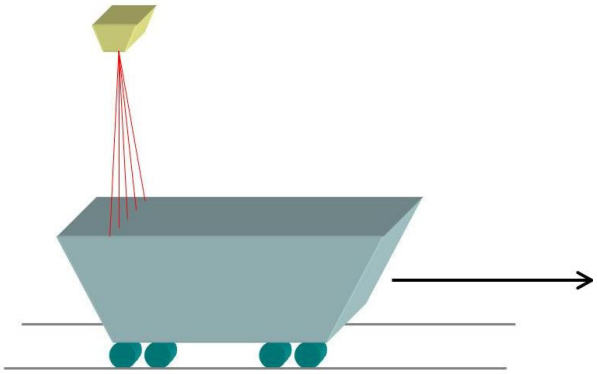


Figure 1.3 – Measuring cross sectional profile.

As the ore car moves along the railway line a scan can be taken at regular time intervals giving a set of cross sectional area values. By integrating the array of cross sectional area values the volume can be determined. To obtain the volume of residual ore the integral can be compared with the integral of a known empty ore car. The difference is found in the same manner as using integration to find the area between two curves.

The volume of residual ore in both the left hand side and right hand side of the cars are calculated as two separate values. The sum of these gives the total volume while the difference gives the imbalance between the two halves. Both these values are scaled by a density value taken from the SCADA system to give the total residual and imbalance quantities in terms of mass.

1.5 System development and testing

This project has been developed and tested using cardboard models. The laser scanner was bolted to a frame supported by the office partitioning as shown below in figure 1.4. Ore car models were scaled to one fourth of the size of an actual car based upon mechanical drawings. These models were constructed from cardboard boxes. The models were given a similar cross sectional profile as shown in figure 1.5. The height of the scanner above the models was also one fourth of the distance expected on a typical installation. Not having wheels the models were kept in line using two lengths of aluminum angle.

Figure 1.4 – Laser scanner workshop testing

Figure 1.5 – Cardboard scale model

Although a simple model setup, it provided an effective test platform. In particular, using the cardboard models allowed the testing of all components in the system. The cardboard models could be filled with objects of known volume to assist in the calibration of the system and provide test data. These models were dragged underneath the scanner using string to represent the cars being ejected from the tipping cell and moving beneath the laser scanner.

1.6 Company Profile

BEC engineering is an electrical engineering consultancy company. BEC is based in Perth with offices in Brisbane and Tanzania. The company has approximately 130 staff consisting of six division managers and several other support staff in drafting, administration and IT support. The management structure is illustrated in figure 1.6 (BEC n.d.).

The power generation team design, commission, audit and supervise construction of power generation systems and power stations. The power distribution team design systems for power transmission and distribution including protection systems, harmonic analysis, power factor correction and earthing. Plant integrity team members conduct assessments for reliability, operational requirements, statutory requirements, safety and risk. A process control division designs multivariable control systems, advanced process control systems, PLC systems and process optimization strategies. The automation division develops PLC and SCADA systems, Operator terminals, telemetry systems and remote operation systems (BEC n.d.).

The company provides total engineering services to the resource, public and industrial sectors. They pride themselves on being an innovative and integrative company. They take care of their various cliental needs and services. Offering a wide range of services and applying their expertise on projects that require design, turnkey solutions, construction supervision, commissioning and or operations support (BEC n.d.).

BEC has an impressive history of working in the Oil and Gas industry at all levels with operators and contractors. In the mining sector BEC has worked on all types of mining projects including gold, copper, nickel, iron ore, coal and diamonds. BEC handles projects in all types of environments (BEC n.d.).

Additionally, BEC engineering provides specialist engineering services in the utilities sector including power generation utilities, independent power producers, customer-owned generation, co-generation, embedded generation and renewable energy companies. BEC also has ongoing involvement with other utility services for water transport and treatment plants, gas transmission pipelines, compressor and delivery stations (BEC n.d.).

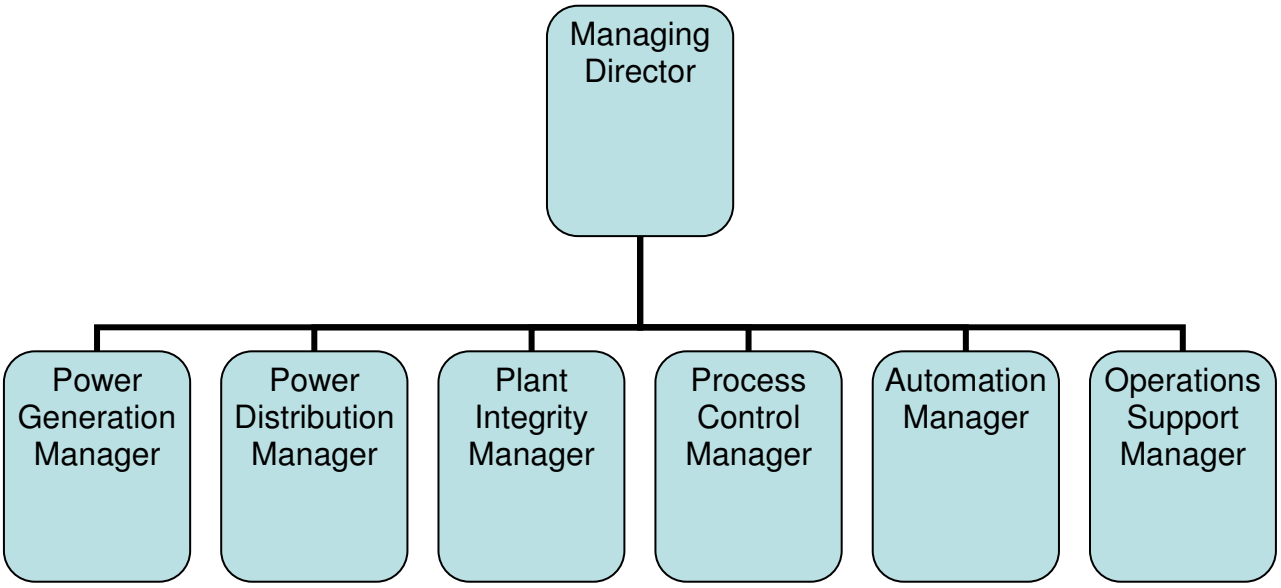


Figure 1.6 BEC Engineering Senior Management

1.7 Project Team Members

The key staff from BEC involved in the project addressed by this report, can be seen in figure 1.7. Overall the project is managed by the Automation manager who developed the initial proposal for the client, developed the scope of work, functional specifications, tender documents, procured the necessary hardware and software components. The lead engineer is the principal point of contact for the client, manages installation contractors, and scrutinizes detail design, supervises technical and engineering staff. The responsibility for cost control, quality management and document management fell upon the project manager. A senior electrical engineer provided detailed design and developed testing procedures. The draftsperson involved in the project provided detailed drawings specifically for schematics and switchboard layouts. The role of the intern was predominantly detailed software engineering for the embedded PC and Citect SCADA, research into effective algorithms, interfacing the equipment, providing technical information to the installation contractors, liaising with the client, assisting with documentation and providing feedback to senior staff.

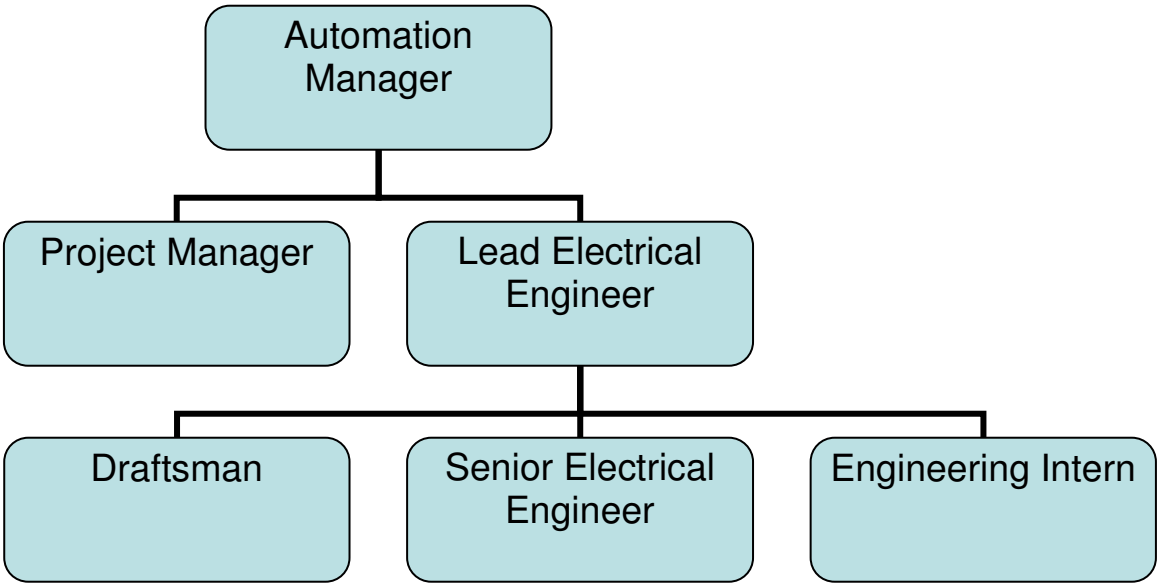


Figure 1.7 BEC Engineering laser scanner project Team

2.0 Laser Scanner

The laser scanner uses the time of flight principal to measure an array of distance measurements through a radial arc. The measurements are then transmitted over serial communications to the embedded PC. Communications connections are made using a 500k baud rate which is not one of the standard Microsoft baud rates and requires a dedicated PCI card. For transmission over the serial communications, the distance data is encased into telegrams. A set of measured values is transmitted after a request is made by the remote devices. Requests are made using a command which is also encased into a telegram.

Figure 2.1 – SICK laser scanner LMS-221

2.1 Theory of operation

The laser scanner chosen for this project, has a fixed laser beam which is reflected by a spinning mirror through a radial arc (SICK 2006b). The unit can be used to measure distances to points through this arc at fixed intervals of angle. The arc size and intervals of measurement depend on the specific model used. The Sick LMS-221 as shown in figure 2.1 was selected and has an arc of 100 degrees. With measurement intervals of 0.5 degrees 201 individual measurements are obtained.

Distance from the scanner to an object is measured using the “time of flight” principal. A pulse of light is transmitted toward an object and the time taken for the light to be reflected back to the unit is measured. The return time is directly proportional to the distance between scanner unit and object as shown in figure 2.2.

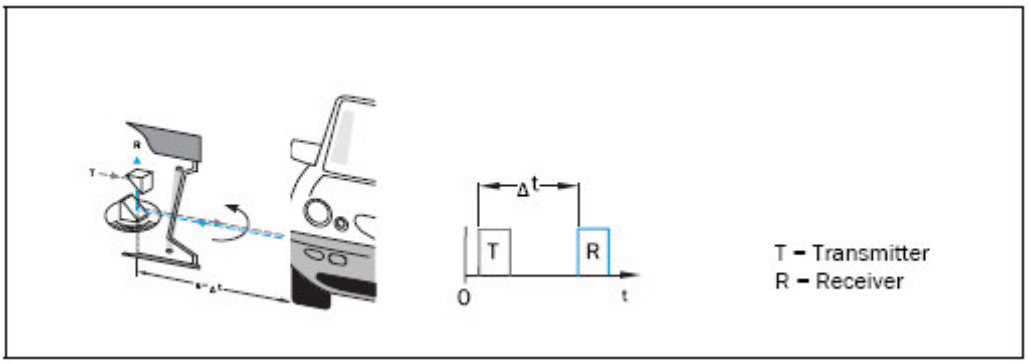


Figure 2.2 – Time of flight principal (SICK 2006b).

There are two modes of measurement available using the SICK scanner. The first default mode will give the measurement resolution in millimetres and a measurement range of 8.1 metres. The second mode has a resolution of 1 cm and a range of 81 metres. In most of the site installations the scanner will be further than 8 metres above the ballast therefore the second mode will need to be used. A change in mode or other configuration settings is achieved using the LMS user software contained on the CD that comes packaged with the LMS unit. This software is also required to change the baud rate for the LMS scanner. A detailed discussion of the selected baud rate used for this project is presented in section 2.3.

Configuration settings can be made permanent by writing to the scanner EEPROM. Once permanently configured using the LMS user software there does not need to be any provision in the car scanning software to change configuration settings. In the event that the scanner will require replacement a procedure for changing the mode and baud rate is detailed in appendix ii.

2.2 Data Communications

The SICK LMS 221 unit can communicate using either RS422 or RS232 serial protocols. RS232 is limited to 10 metres while RS422 can go as far as 1200 metres (SICK 2006b). In all the installations the distance from the laser scanner to the equipment room is greater than 10 metres and therefore RS422 communications will be used. RS 422 transfers data using two balanced pairs which has better tolerance to noise allowing greater cable distances and faster data transmission (Lammert Bies 2008). SICK (2006b) also specifies that cables must be screened twisted pair. To meet this specification a category 5 foil screened twisted pair cable has been selected and installed.

2.3 Baud Rates

The SICK laser scanner has a choice of four baud rates 9600, 19200, 38400 or 500 000. The selected baud rate must be sufficiently high enough for all the data from a scan to be transmitted between the requests for scans. Should a request for values be made before the completion of a previous scan then the transmission is aborted resulting in an invalid telegram and 100% loss of data. Minimum scan intervals were tested for each baud rate with the results shown below in table2.7.

Baud Rate	Minimum scan Interval (ms)
9600	1000
19200	500
38400	250
500000	75

Table 2.7 - Minimum scan intervals.

The baud rates 9600, 19200 and 38400 are standard baud rates (Microsoft 2008) and will work with a standard PC serial communications port. To enable serial communications at 500 k baud a specifically manufactured serial port and software capable of communicating at high speed is required. 500k baud is much faster than the traditional serial speeds and is also a non standard baud rate. Communication at 500k baud requires a serial PCI card that is manufactured to SICK specifications. (SICK 2006b)

This project is coded in Visual Basic 6 which uses the Microsoft Active X object MSCOMM32 for serial communications connections. MSCOMM32 will only accept serial connections at the standard baud rates 9600, 19200, etc up to 115 k (Microsoft 2008). To enable serial communication connections at higher speed the SICK specification PCI card has provision to multiply the clock rate. Over clocking is achieved by closing the X8 jumper on the PCI card as shown in figure 2.8. This jumper sets the clock speed to 8 times the speed specified by MSCOMM32. This allows the MSCOMM32 object to think that a slower clock speed is being used. For example, with MSCOMM32 baud rate set to 56000 and with 8 times clocking enabled gives a true serial baud rate of 461 000 even though 56000 is an acceptable baud rate to the MSCOMM32 component.

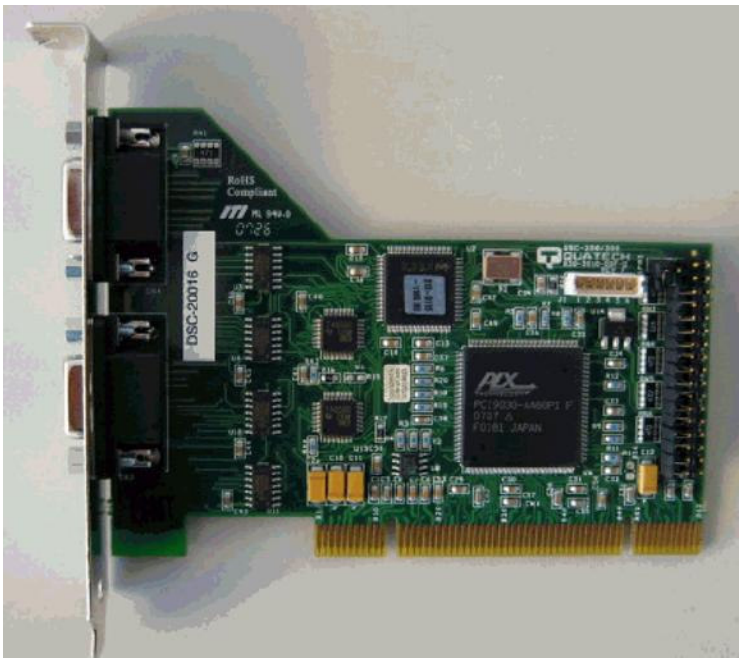


Figure 2.8 - High speed serial Communication PCI card.

A standard serial port UART is driven by a crystal controlled clock speed of 14.7456 mhz. The 500k serial port developed for the SICK unit has a clock speed crystal frequency of 16.000 mhz. The slightly higher frequency for the clocking frequency will now give a true data rate of 500 000 baud. With the combination of using a 16 MHZ clock with the times 8 jumper closed and the MSCOMM32 baud rate set to 56 000 allows the PC to communicate with the laser scanner at 500 000 baud. A table of baud rates for the high speed communications card is shown in table 2.9 (SICK 2002).

Target baudrate →		**9600**	**19200**	**38400**	**500k**
Use baudrate value	MOXA and Quatech card	1075	2212	4300	56000
	PCMCIA SIO-RS485-SICK	2150	4301	8602	115000

Table 2.9 - Baud rate conversion table. (SICK 2002).

2.4 Retrieving scan values

The data format used in communications from the laser scanner takes the form of telegrams. Each byte has 10 bits with 1 start bit, 8 data bits and 1 stop bit. The bytes are assembled as blocks. Control command blocks are sent to the scanner or response blocks are received from the scanner. These blocks of data sent to the scanner are referred to as *send telegrams* while the blocks received back from the scanner are *receive telegrams*. During normal operation the laser scanner will send a response telegram in reply to a command telegram. The structure of a send telegram is contained in table 2.10. The structure of a receive telegram is contained in table 2.11. The send telegram is always 8 bytes long while a receive telegram can be up to 518 bytes long (SICK2006a).

Referring to table 2.10 and table 2.11, the frame components of the telegram are the same in both the send and receive telegrams. The start of text is the value 02h which is a preamble to identify the start of the transmission. The address byte is the bussable address which is not used on this model and is 00h by default. The length values are the number of subsequent data bytes not including the check sum. Byte 5 is the command / response byte with the associated data placed into byte 6. The checksum is used for error detection for which the algorithm is detailed in SICK (2006a)

In this project the scanner has been pre-configured using the SICK *LMS-400 user software* (SICK 2006c). Therefore, the embedded PC will generally only require commands for the normal day to day operation but not configuration commands. Specifically these commands will be to request scan values from the scanner and to turn the laser on and off. SICK (2006a) contains a detailed discussion of a large number of telegrams which can be used to communicate with the laser scanner.

	Frame			Commands & Data		Frame
Byte Position	1	2	3&4	5	6	7&8
Description	Start of text	Address	length	Command	Data	Checksum

Table 2.10 - Command Telegram Structure (SICK 2006a)

	Frame			Response & Data		Frame
Byte Position	1	2	3&4	5	6 to n	N+1 & n+2
Description	Start of text	Address	length	Response	Data	Checksum

Table 2.11 - Response Telegram structure (SICK 2006a).

2.5 Send telegrams

Two different strategies were assessed for retrieving scan data from the laser scanner. The first strategy is based upon Rodgers (2007) who used the command 30h-01h to request a single array of scan measurements. This telegram is constructed as a string and sent to the scanner using the MSCOMM32 Active X component included as part of the visual basic package. Using the visual basic event timer the telegram is sent at regular time intervals. The response from the scanner is one telegram containing an array of measurements.

The algorithm developed by Rodgers (2007) was for 9600 baud and failed to work properly at 500 000 baud. The software on the embedded PC waits until it receives an array of scan measurements from the laser scanner, once a request for measurements has been made. Sometimes there was no response from the scanner. This problem was investigated using an oscilloscope which showed that the command was simply ignored by the laser scanner. The send telegram contains a checksum used by the scanner to test the integrity of the telegram (SICK 2006a). It is likely that the data was corrupt; however this could not be proved.

The second strategy used was to retrieve the measured values based on the algorithm discussed in SICK (2002) and used by Drexel (n.d.). This algorithm used the command 20h-24h to request measured values continuously. This command will cause the scanner to send arrays of data one after the other without waiting for a request. The interval between scans depends on the baud rate selected. The scanner will send the next telegram as soon as the previous telegram is finished.

Using the continuous mode means that all scans can be retrieved but the scan rate cannot be controlled. At 500 000 baud the interval between telegrams is approximately 75 milliseconds. This project requires a scan interval of approximately 200 milliseconds therefore we must discard every second or third scan. The variation in scan intervals will cause an error in the integration calculations however a high degree of accuracy is not required in this application. To achieve greater accuracy there is a command which will return the measured values with a real time index. This index could be used as the third dimension for real time scanning applications.

2.6 Receive Telegrams

The structure of the receive telegram is shown in figure 2.4. To be able to convert the received signal to a set of values the telegram has to be dissected. A flow chart of this process is shown in figure 2.12. In this project the telegram interpretation is achieved using the algorithm developed by Rodger (2007). This algorithm is a stable and industrial strength method in taking the results from the raw data. This algorithm periodically polls the Active X object MSCOMM32 and extracts the data from the objects buffer. A search is made through this data for the telegram header and any unwanted data is discarded leaving an array of polar co-ordinate results.

An alternative method of interpreting receive telegrams can be seen in Drexel (n.d.). This method is based on the same general flow chart as Rodger (2007) but uses the event generator in MSCOMM32. When the MSCOMM32 buffer size exceeds a preset threshold the object will call an event function.

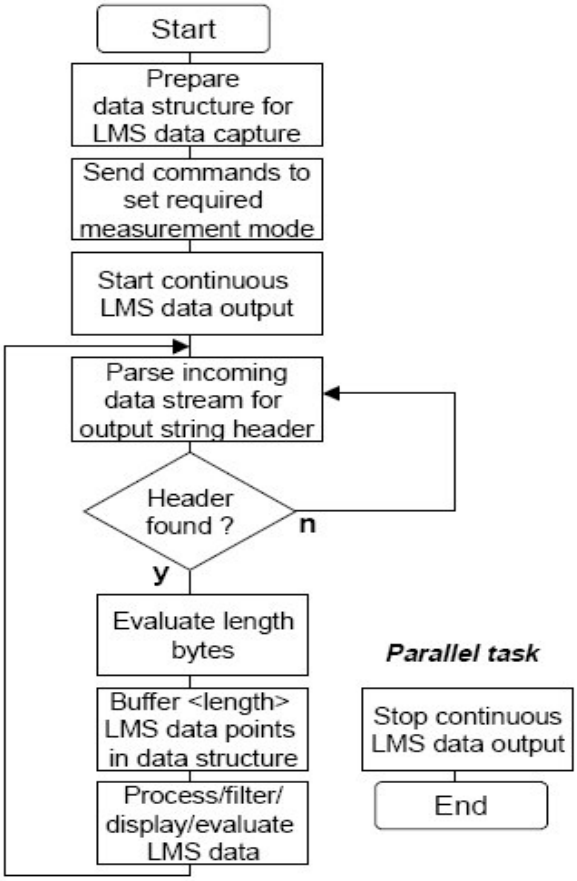


Figure 2.12 – Flow chart for receive telegram interpretation SICK(2002).

2.7 Turning the laser on and off

In order to preserve laser life the laser will be turned off when there is no train being unloaded. The electronics will be left on permanently to prevent condensation forming on the laser window. The laser scanner also takes approximately 2 hours to reach thermal equilibrium after it has been turned on (SICK 2006b). Turning the laser on and off can be achieved using the data communications connection. Command 68h-01h is used to turn the laser on while command 68h-00h can be used to turn the laser off.

3.0 Embedded PC

Raw measurement data is passed from the laser scanner to the embedded PC where the data is processed. The raw measurements are an array of polar co-ordinates which first have to be converted to cartesian co-ordinates. Search algorithms have been developed to find the ends and sides of the wagon. Integration of each cross sectional profile determines the cross sectional area. The array of cross sectional areas is then integrated to find volume. The embedded PC also contains databases for data on each type of wagon, results, raw scan data and a table of set points.

3.1 Hardware and operating system

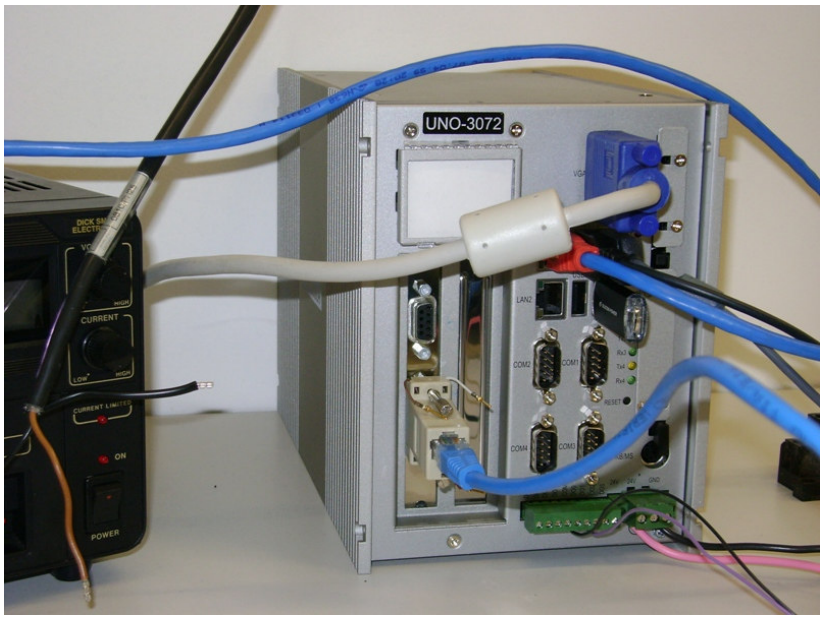


Figure 3.1 embedded PC with RS422 serial PCI card.

At the heart of the system an embedded PC has been used to perform all the necessary calculations. This PC will interface between the laser scanner, existing PLC's and Citect SCADA systems as previously shown in figure 1.2. The PC will also be used to store the scan records as a database. An embedded PC was chosen in preference to a PLC due to the complexity of the calculations and the need for a database. This rugged PC has been chosen to suit an industrial environment and has a low power 900 Mhz Intel Celeron Laptop processor. It has most of the general features as a standard desktop PC, a low power graphics accelerator, USB ports, serial ports and facility for battery backup power (Advantech. 2007).

The operating system installed on the embedded PC is Windows XP embedded. This is a condensed version of windows XP (Advantech. 2007). This operating system will fit on the flash memory removing the need for a hard drive (Advantech. 2007). However, a 160Gb hard drive has been fitted for the purpose of storing the database. The database does not contain crucial information and the PC does not require any backup facility. The PC has also been fitted with the 500mhz Quatech RS422 serial PCI card to communicate with the laser scanner.

3.2 Processing the results

Once the laser scanner has transmitted a set of polar co-ordinates to the embedded PC the values need to be interpreted. The algorithm for this interpretation is shown on a flow chart in appendix iv. Data is received from the scanner through the serial connections and converted to an array of integers as detailed in section 2.6.

The scanner head is not always exactly at right angles to the ground. On some sites the scanner has to be tilted to avoid obstructions and to have full line of sight. In these cases the scan data needs to be scaled down by the cosine of the tilt angle as shown in figure 3.2. This tilt angle is read from a text based configuration file when the program is started.

Height of the scanner unit above the ground is required to convert the array of polar co-ordinates to the array of Cartesian co-ordinates relative to ground level. This height value is also read from the text based configuration file.

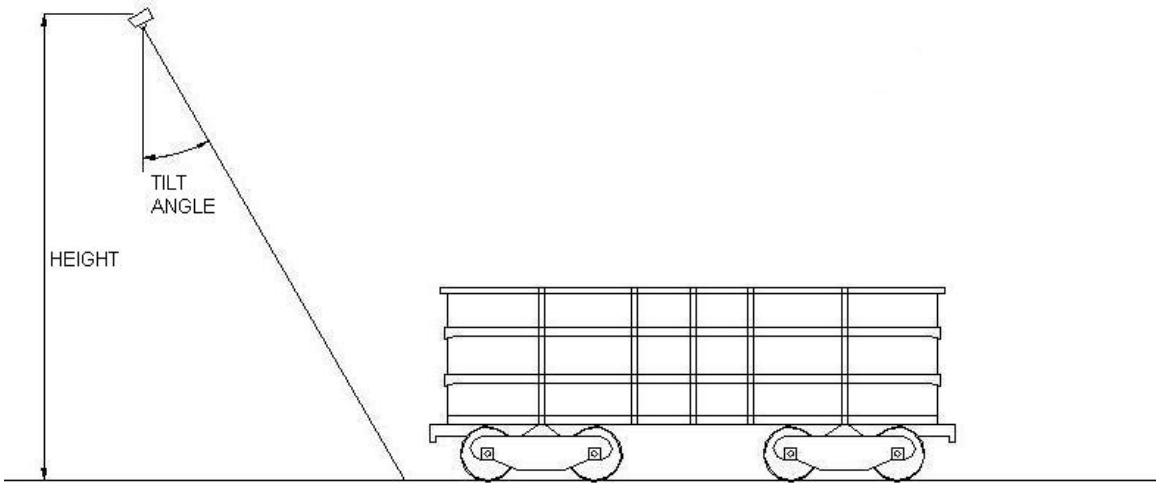


Figure 3.2 – Scanner calibration parameters tilt angle and height.

The laser scanner measures all the points over a radial arc of one hundred degrees. In most installations the scanner is located well above the height of any ore cars and therefore only a small part of the arc is actually used for the measurements. To ensure the integrity of the calculations, minimizing computational effort and restricting the field of view available in the SCADA/HMI, the unwanted results must be removed. A typical installation will only require measurements over a thirty degree arc. In order to filter unwanted data the software requires an arc size in degrees for the scanning window as seen in figure 3.3. This value is normally large enough to see the whole ore car with a bit extra to allow for variations in the cars. This scan window angle is also read from the text based configuration file when the program is started.

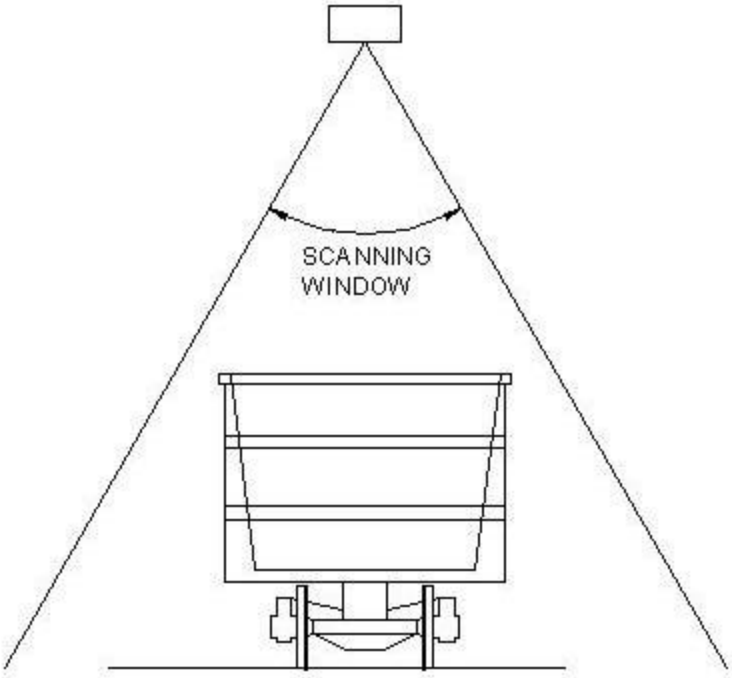


Figure 3.3 – arc angle required to scan the ore car

The next step is to convert the polar co-ordinates to a set of Cartesian co-ordinates. The polar co-ordinates are contained in an array where each value represents the length from the scanner to an object. The polar angle is implicit, where the angle value is taken as the array subscript noting that these values are all measured 0.5 degrees apart representing -50 degrees to +50 degrees in the two dimensional polar plane. The following equations give an array of Cartesian X values and an array of Cartesian Y values from the array of polar co-ordinates.

$$X_n = Polar_n Sin\left(\frac{\pi(n-100)}{360}\right)$$

$$Y_n = Polar_n Cos\left(\frac{\pi(n-100)}{360}\right)$$

Where; X_n is the n'th Cartesian X value in the array
Y_n is the n'th Cartesian Y value in the array
$Polar_n$ is the n'th Polar length value in the array
n is the array index

Once a set of Cartesian values are determined, the next step is to invert the Y values. The co-ordinates are initially relative to the scanner therefore to make the points relative to ground and positive the height of the scanner above the ground needs to be factored into the equations. The height from the scanner base to the ballast is typically taken at the base of the railway line as shown in figure 3.2. This height value is read from the text configuration file at start up. The individual Cartesian Y values are subtracted from the height value to give the Cartesian values relative to ground level instead of relative to the scanner.

3.3 Triggering

To be able to determine an accurate measurement for the volume of residual ore the PC must know when to integrate the scan data. Integration needs to be performed only when an ore car is under the scanner and the car is moving. At the site installation information obtained from the PLC registers can be used to determine a suitable start and finish time. For the initial development, using the test model platform triggering was achieved by using a search algorithm.

Once a set of Cartesian values are calculated the results are then examined to determine if an ore car is present. Several simple algorithms were tested for the identification of an ore car. The first algorithm compared the mean Y value with a set point but was prone to false triggering. The second algorithm searched for the outsides of the ore car by looking for values of y that were the equal the height of the car within a defined range of X values. This algorithm improved on the first but was triggered by the ends of the ore car. The final algorithm also searched for the inside of the car wall. If the search tool found a left side and right side and a left inside and a right inside then the integration would proceed. The final algorithm was still prone to false triggering but was sufficient for workshop testing.

3.4 Finding the profile

Once the Cartesian co-ordinates corresponding to the ore car have been ascertained the next step is to integrate. Selecting a suitable point to begin and end the integration on the cross sectional profile has also been researched. Different strategies have had a significant impact on the accuracy of the final answer.

In this project one of the key aims is to be able to determine the degree of imbalance between each half of the car. To determine this we required separate values for the volume in each side. To calculate the two values for volume, integration needed to be performed over each side of the car. Firstly, integration can be made from a side wall to the centre point and secondly integration is made from the centre to the other side wall.

The first integration algorithm trialed, calculated the integral between fixed points in space. Each car half was integrated from the centre of the measurement array at zero degrees to a point in the array which was just outside the ore car. This algorithm produced excessively large variations in results. The cardboard models had sideways free-play between the railways which most likely caused a variation in results. Integration needs to follow the ore cars in order to integrate between fixed points on the car instead of fixed points in space. An algorithm had to be developed to find a reference point on the cars which would be found consistently in the same place in order to produce consistent results.

A separate program was developed to simultaneously trial five different methods of searching for a point which could be accurately reproduced. This program started from the outer most scan result and searched through the data looking for the side wall of the car. The five methods were:

1) the first Y value to exceed its set point
2) the first delta Y value to exceed its set point
3) the first derivative of Y to exceed a set point
4) the first delta Y value to exceed a negative set point
5) the first derivative of Y to exceed a negative set point

This program was allowed to run for an hour while scanning a stationary object to determine if any of the methods produced a consistent result. A sample of this test data is contained in appendix iii. While some worked better than others none were consistent enough to be used as a reference point. Further investigation revealed the reason for the inconsistency is that the light was hitting the edge of the object and causing an inaccurate distance value. This problem is referred to in SICK (2006a) as the edge strike or halo effect.

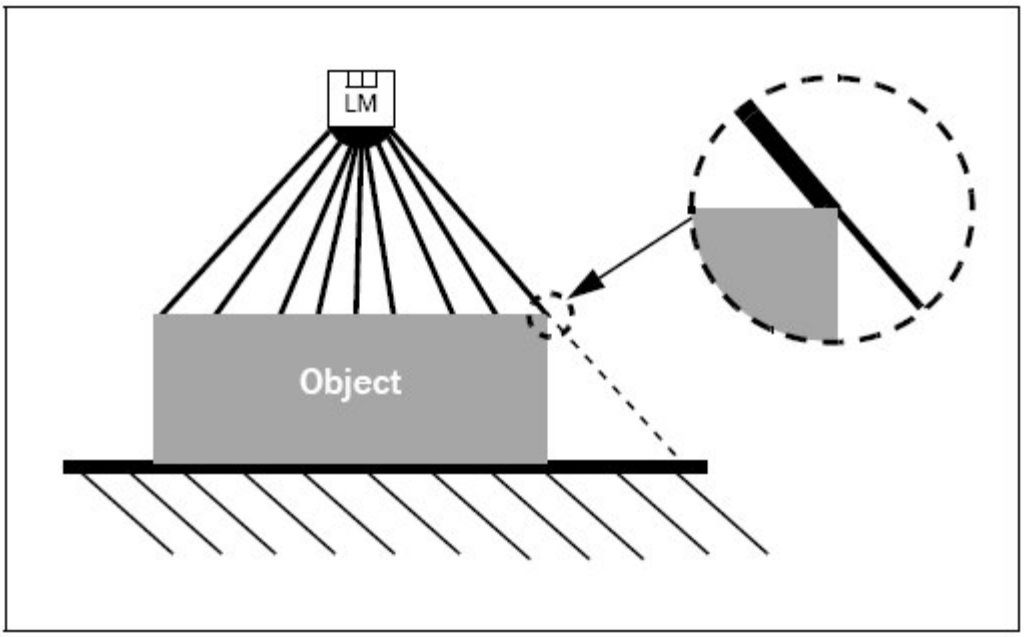


Figure 3.4 – Edge strike halo effect (SICK 2006a)

Even though a laser is being used, this laser is of a low intensity type 1 laser and has a very wide diameter spot (SICK 2006a). Table 3.5 is a table showing the spot diameter at various distances from the scanner.

Distance in m	Spot diameter in cm[1)]		Spot distance in cm[2)]		
	LMS200/220	LMS211/221/291	0.25°	0.5°	1°
1	2	3	0.5	0.9	1.8
2	3	4	0.9	1.8	3.5
3	3	6	1.4	2.7	5.3
4	3	7	1.8	3.5	7.0
5	4	8	2.2	4.4	8.8
6	4	9	2.7	5.3	10.5
7	5	10	3.1	6.2	12.3
8	5	12	3.5	7.0	14.0
9	6	13	4.0	7.9	15.8
10	6	14	4.4	8.8	17.5
15	8	20	6.6	13.1	26.2
20	10	26	8.8	17.5	35.0
25	13	32	11.0	21.9	43.7
30	15	38	13.1	26.2	52.4
40	19	50	17.5	35.0	69.9
50	24	62	21.9	43.7	87.3

Table 3.5 – Laser spot diameter. (SICK 2006a)

In the event of an edge strike where only part of the beam hits the object an incorrect measurement will result. The algorithm inside the laser scanner adjusts the raw time of flight measurement by a factor according to the intensity of light reflected back from the object (SICK 2006a), according to ;

Output measured value = raw measured value+ distance correction in accordance with the internal distance table + correction in accordance with the internal received energy table. (SICK 2006a)

This calculation assumes that there is always a full spot strike (SICK 2006a). In the event of an edge strike the raw result will be scaled incorrectly and an incorrect measurement will result. The measurement could be either closer than the object or further away (SICK 2006a). Figure 3.6 shows a cross sectional profile which contains a co-ordinate affected by edge strike. These edge strike values are random in nature and a similar effect on the system as noise from any instrument. An algorithm had to be developed which made the calculation results immune to the effects of this random value. The starting and finishing points must be inside any points affected by edge strike and be consistently reproducible.

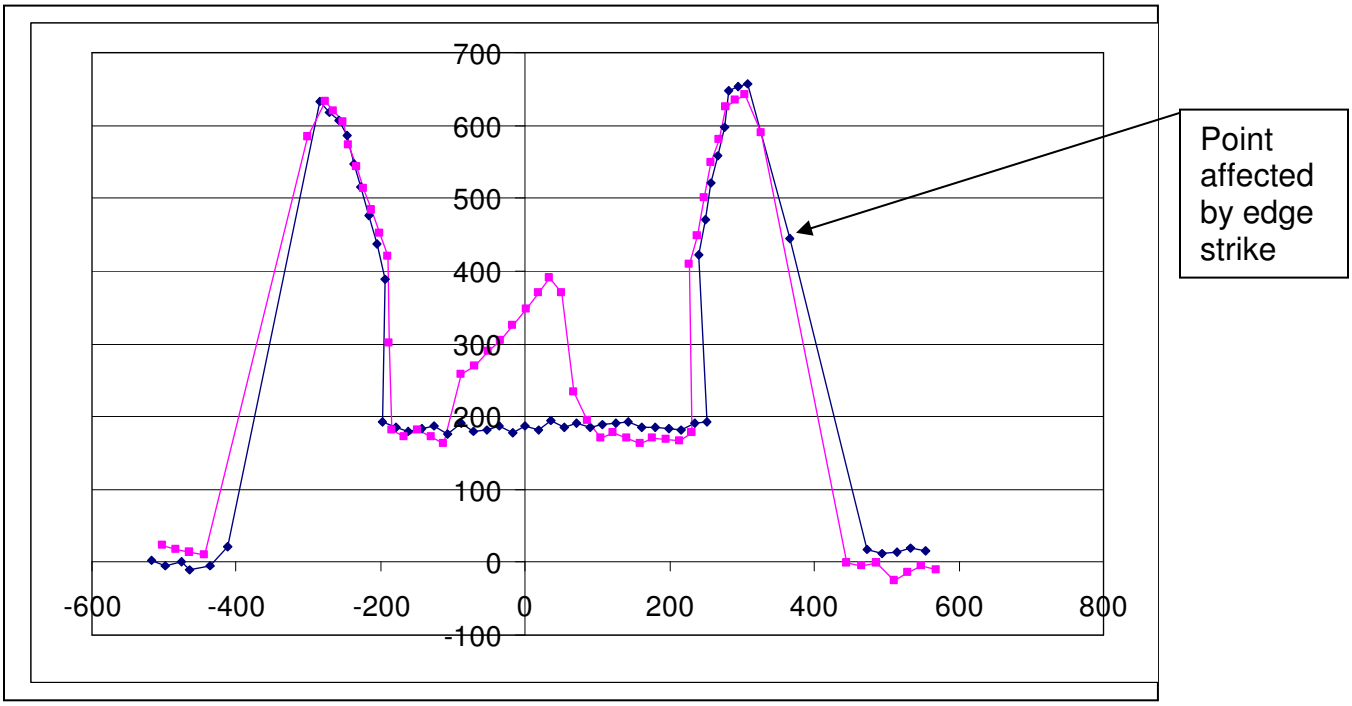


Figure 3.6 - Cross Sectional profile.

Three algorithms were developed and tested for consistency and immunity to edge strike. The first algorithm tested, searched for the top of the side walls within a fixed window as shown in figure 3.7. The algorithm found the mean X co-ordinate for all the Y co-ordinates above a predetermined set point over the entire length of the car. This gave an X value representing the top and centre of the wall which could be consistently reproduced from one car to the next to be used for a reference and was also relatively immune to the effects of edge strike. Finding an X co-ordinate representing the centre of the car is now a simple matter of finding the value half way between the two side walls. Y co-ordinates for the two sides X values and the centre value are found from linear extrapolation of the neighbouring values.

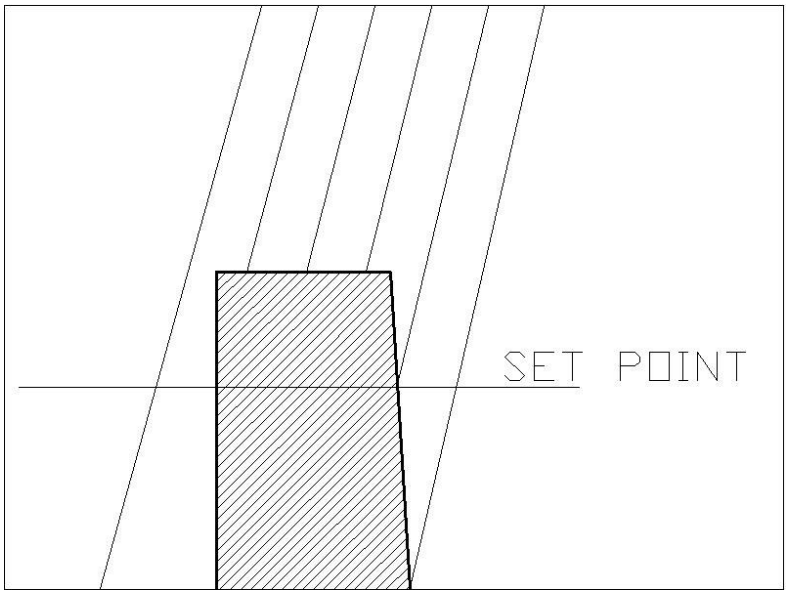


Figure 3.7 – finding the side wall

The second algorithm which was developed used linear extrapolation to find the outside of the car. This method searched through the scan data to find where the profile Y co-ordinates crossed the set point on the outside of the ore car. Linear extrapolation was used to find the X value that would correspond to the set point from the two adjacent points from either side of the set point. This X value could now be averaged over the first quarter of the car, reducing the effect of instrument noise. The calculation is repeated for the second side.

These two algorithms were compared by scanning a stationary object while watching for variation in results. The raw test data can be seen in appendix v. This illustrates that the second algorithm has far less variation in final value with less than 1 millimetre while the first algorithm has approximately 3 millimetres variation. It was also noted that the first algorithm exhibited incorrect values when the contents of the car exceeded the set point height. The second algorithm was vulnerable to changes in height of the side wall. Because the X value is extrapolated from known Y co-ordinates, a change of Y value has a significant impact on the extrapolated result.

The third algorithm tested was a combination of the two previous algorithms. This algorithm searched through the data to find the furthest most point from the centre that exceeded the set point. The mean of this value and the two adjacent inner values was used to calculate the reference value for X. This method was found to be more immune to changes in the height of the side wall and furthermore able to produce reliable results where contents of the car were higher than the side wall. While the final result did not determine the X co-ordinate as accurately as the previous algorithm it was selected as the most suitable candidate to make the system as robust as possible.

3.5 Profile Integration

Two integrations are performed to convert the data arrays to volume. The first integration finds the cross sectional area of the ore car for each scan. This integration is performed numerically using the trapezoidal rule. This rule has a good level of accuracy well beyond the requirements of this application. The trapezium rule is also easy to convert to a form that can be used where the ΔX values are not equal. Because the co-ordinates are converted from an array of polar values the ΔX between points has a substantial variation. This variation is too large to ignore. According to Croft (2001), the integral is the sum of individual areas of trapezia where :

$$AreaOfTrapezia = \frac{\Delta x}{2}\left(y_{n+1} + y_n\right)$$

Easily adapted for changing ΔX values to;

$$CSA = \sum \left(x_{n+1} - x_n\right)\left(\frac{y_{n+1} + y_n}{2}\right)$$

A second integration is then performed to find the volume. In this case we introduce the third dimension. The scans are taken at regular time intervals as the ore car passes underneath the scanner. If a constant velocity is assumed then this third dimension can represent the approximate Z co-ordinates. Because there is no definite start and finish point we can no longer use the trapezium rule. Instead we have to divide the length of the car into arbitrary ΔZ values and assume that the scan has been taken in the middle of each division. This method is the same concept as the midpoint rule for integration (Stewart 1995). The ΔZ value is determined from the known length of an ore car divided by the number of scans taken over the car being scanned.

Using the triggering algorithm detailed in section 3.3, the number of scans taken for each car that passes is counted by the embedded P.C. and is not assumed to be equal. The only assumption that could cause incorrect volumes is the assumption that velocity is always constant. The actual velocity value is not required, however increased velocity will result in less scans per car and lower resolution. The length of the ore car is recorded in the database containing the information on each car type and retrieved as required. The length divided by the number of scans gives us an equivalent to a ΔZ value. Volume is now calculated to be:

$$Volume = \frac{CarLength}{NumberOfScans} \sum CSA_n$$

The volume of residual ore is the difference between integral of the scanned ore car with the integral of an empty ore car. These values are stored in the database for each type of ore car and retrieved with the ore car length when required. Integral values for empty ore cars will be determined by statistical information obtained from the results database once the system is installed on the car dumpers.

Separate values of volume are calculated for each side of the ore car. The sum of the two volumes gives us the total volume which is multiplied by the density to give a total mass of the residual ore. The value of density is obtained from the set point database on the embedded PC. This value can be altered by operations staff using SCADA/HMI. An absolute value of the difference between the volumes on either side multiplied by the density gives us the tonnage imbalance.

3.6 Alignment

The eventual aim for this equipment is to use the quantity of residual ore to invoke a procedure to re-tip an ore car. A re-tip decision should only be made on condition that the car is still on the rails and has not been damaged during the tipping process. There has been an incident where the bogies of the rail ore car, had been completely broken away from the car during the tipping process. If a car has derailed or a bogie damaged when it is pulled back into the tipping cell there will be insufficient clearance between the cell arms which hold the ore car and the car itself, resulting in damage to the tipping cell. An algorithm had to be developed to confirm that a car is aligned within the tipping cell arms before being pulled into the cell.

3.6.1 Skew

In order to confirm the car is on the rails after it has been emptied, an algorithm was developed to determine if the top of the car is parallel to the rails. An early alignment algorithm used the same search tool as the integration algorithm to find the centre of the top of the wall. The scans over the first quarter of the car were used to find an X co-ordinate representing the centre of the car at the front. A second value was found for the rear of the car. The values were compared and scaled by 4/3 to give a numerical figure representing the difference between the front and rear of the car relative to the track. Testing this algorithm found the required level of accuracy could not be obtained. This method was also inaccurate where there was a large amount of ore in the car. A third problem found was that in the event where the entire car had been derailed, the car could still be parallel with the rails but the derailment would not be detected.

Two changes were made to the alignment algorithm in order to address these shortcomings. The first change removed the search for the inside edge thus preventing problems when the car contains ore. Instead of being compared to the rear of the car the second change involved this value being compared with a fixed point in space to determine if the car would hit the tipping cell when the car was pulled back in. This algorithm was repeated over the last quarter of the car. The results had to be taken as an average of multiple scans in order to reduce the effect of instrument noise. By comparing the value with a fixed point, the error would be immediately halved. The output value for this algorithm is then the difference in centimetres in X value between the centre on the car and the centre of the rails. This value represents the points one eight from each end of the car.

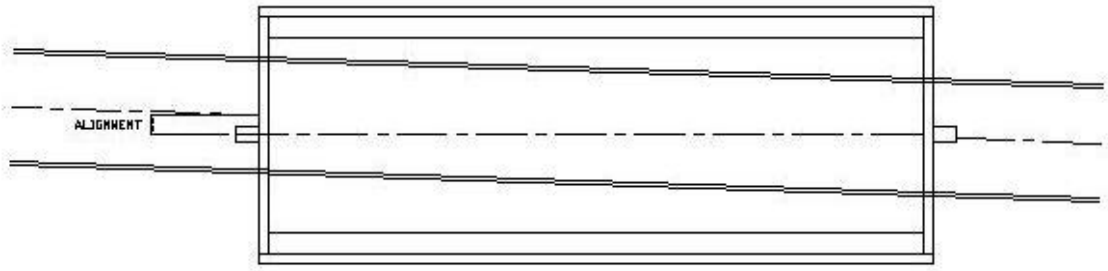


Figure 3.8 – Car Alignment

3.6.2 Level

Another algorithm which was considered to help determine if a car is correctly aligned is to compare the height of the side walls. This algorithm searches through all scan results for any y values above the set point. By taking a mean of all those on one side of the car and comparing this value with the mean of the second side of the car a difference in height is found. This difference represents degree of tilt and can be used to help determine if the car is on the rails or if the car is not sitting on the bogie correctly. This algorithm is reasonably robust but can't distinguish between the car and the ore in the event that the ore in the car is higher than the side wall of the car.

In the acceptance testing this algorithm will be tested by chocking up one side of the car. The car body is not permanently attached to the bogies and relies only on gravity to hold the body down in its correct position. In order to test if the scanner will detect the event that a car is not on the bogie correctly, the car will be jacked up and chocks placed between the body and the bogie to simulate the situation where the body is not correctly sitting on the bogies. The output level value is the mean of the difference between the side walls heights measured in centimetres at the points shown in figure 3.9. The ore car bodies are very rigid and not prone to twisting. Variation of level from one end to the other is very small.

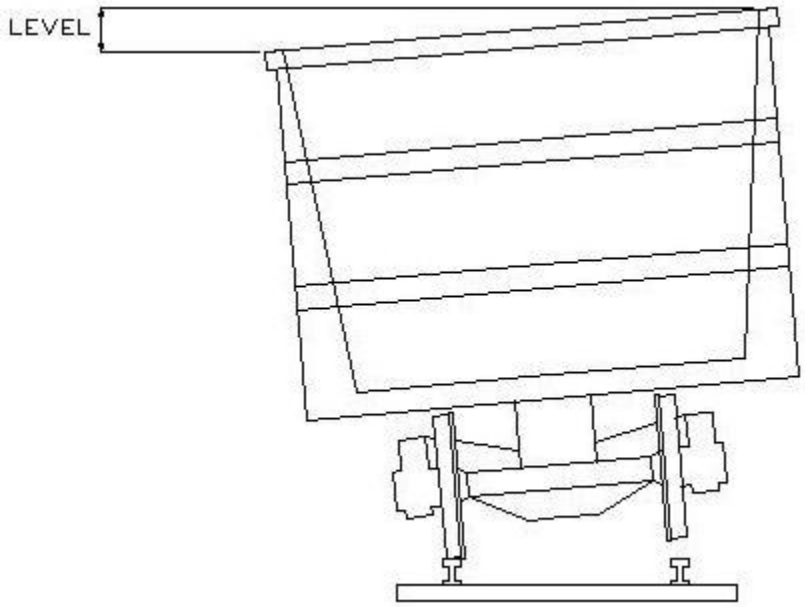


Figure 3.9 - Car level.

3.7 Database

Included in the scope of works for this project was the facility to store scan results in the form of a database. Initially this database will be stored on the embedded PC and may be moved to a file server in the future. The local database contains four tables. The first table contains data for each type of ore car. The second table contains the results for each car scanned while the third table contains raw scan results for any car which tripped an alarm. The fourth database will store set-point and other variables. All results data will be kept for 60 days before being deleted automatically. The SCADA system however will only have access to data on a train while it is being unloaded.

Table one titled *profiles* contains information about each type of ore car that is owned by the client. The *length* field is the measurement of length inside the car body in centimetres. Length is required to calculate the integral of the cross sectional areas as discussed in section 3.5 . The fields *LHS volume* and *RHS volume* are the cross sectional volume integrals of empty ore cars. These values represent the volume of an empty ore car as measured by the scanner and are subtracted from the volume determined by the scanner to give the volume of residual ore. The procedure for obtaining this value is to set the database value to zero and then scan multiple ore cars that are known to be empty. The system will record the volumes of these ore cars in the results database. A suitable average value can then be used as the reference volumes. The two values do not have to be identical. They will only be expected to be the same only when the laser scanner is exactly level. This table also contains a series of X and Y co-ordinates. These values are for profiling different car types in a future development.

The second table named *results* contains the information calculated for each ore car after it has been emptied and scanned. A sample of this table is contained in appendix vi. The *results* table contains the following fields:

1) date & time the results were calculated
2) car ordinal within train
3) car type
4) volume of ore in left side
5) volume of ore in right side
6) the tonnage of the residual ore
7) the difference in tonnage between each half
8) car level
9) car front alignment
10) car rear alignment

A third table titled *scandata* contains arrays of raw polar co-ordinates for each scan where the ore car triggered an alarm. This information is necessary so that if operation staff wishes to visually see the car which triggered an alarm, the information can be posted onto the SCADA network. Each individual scan is kept as a separate record with a field *point_num* that contains an ordinal value for each record. The records are grouped by another field named *time* which contains a string with the date and time the alarm event occurred. The time value is used to uniquely identify one car from another and used to retrieve a recordset of scan result records when requested. The time value was used because it is the only variable unique to each alarm event. The car ordinal value is duplicated when a subsequent train passes through the dumper. The records are placed in the correct order using the field *point_num*.

The fourth table stores the system set point values. This table only has two fields which are titled *tag name* and *value*. The *tag name* is simply a name for the variable as a string. All the values are of type double. These values are stored in a database instead of being stored as local variables in RAM. This means that these values are stored on the hard drive so that the values can be retained when the system is restarted or if the power to the system is lost. These values were kept in a database format in order to keep the program structure consistent.

3.7.1 Database Software Functions

Microsoft Access was used to build the database template and structure. Visual Basic 6 has a number of tools to work with this type of database. This was the motivation behind using visual basic as the programming language. The project makes use of the active X data object ADO developed by Microsoft and uses the "JET" engine. This object can be used to access data on a number of types of databases including SQL Server, Oracle, Excel and internet based ASP (Connell 1998). If the database is ever moved to a file server then this code should facilitate the transfer.

Six of the functions in the visual basic code use the ADO object to communicate with the access database. These functions will require changes when the database is moved to the network server. A brief description of these functions is given below.

The function Loadprofile() uses an SQL search to find records for the car type. This function is called every time a car has been scanned to obtain the length and reference volumes which are returned by the function as a user defined data type object.

The function saveResults() is called each time a car is scanned after the calculations have been made. This function creates a new record and saves all the calculation results.

The function saveScanData() is called when a car triggers an alarm. This function creates a new record for each scan and saves the array of polar co-ordinates together with an index value for the order of the scans in the car and the time that the alarm was generated.

The function ShowAlarmResults() extracts the results from the database and loads them onto an Active X SCADA communications object. The Active X object emulates a PLC and is discussed in detail in section 4.1. This function uses the time the alarm occurred to execute an SQL search to find the correct record. The results are directly loaded onto the registers to be read by the SCADA. This function is discussed further in section 4.4

The ShowAlarmData() function retrieves the polar co-ordinates from the database. This function uses an SQL search to find a recordset which contain a set of scans for an ore car which set an alarm. A recordset is found using the date and time the alarm occurred as the unique identity. Records are placed in correct order by using their ordinal value. Values are placed onto the SCADA Active X registers six arrays at a time. The function keeps track of which arrays have been placed for collection by using a pointer. Each time the function is called it sends the next six polar arrays. The function is called by CheckAlarm() at intervals from the timer if the last arrays have been collected and there is more arrays to be posted.

Every time a train has finished being unloaded the cleanup() function is invoked. This function examines the results table and scan data table. The tables are searched to find records more than sixty days old. These records are then deleted permanently from the database in order to keep the database to a manageable size.

3.7.2 View database mode

When a request for alarm records is received from the SCADA system the embedded PC sets a flag to go into the historic mode. In this mode the scanner results are prevented from being placed into the SCADA communications Active X object registers. System outputs and graphics will be prevented from being shown on the Citect GUI. All other functions will continue normally. The laser scanner will continue to examine the ore cars and generate more alarm events if necessary. When an alarm event is generated all the scanner results and scan profiles are recorded into the database with the date and time when the alarm was generated. The alarm date and time is also stored into a local variable array. When alarm results are requested the embedded PC refers to the date and time corresponding to the appropriate alarm number entered into a field by the citect operator. The embedded PC executes an SQL search to find the results for the requested alarm. Once these are found the results are placed directly into the SCADA Active X communication object registers.

3.8 Alarm Loop

The embedded PC software verifies communications with all other associated equipment at regular time intervals. In the event of failure in one of the system components operations staff are notified via the Citect SCADA system. The alarm loop scans communication from the laser scanner, database, PLC and the Citect SCADA system. A flow chart in appendix vii shows the alarm loop logic. In the event that communications is not received from one of the devices a fault alarm will be generated which can be seen by the operator in the control room. For record keeping purposes the nature of the fault is recorded into a text based event log file . A sample of this file is in appendix viii.

3.9 PLC Communications

To integrate the laser scanner system into each of the existing site's communication links had to be established with the existing site PLC's. Information required from the PLC's include whether a train was present, when the cars were being ejected from the tipping cell and the ordinal number of the car being ejected. The PLC's are connected to an Ethernet network for SCADA communications thus providing a straight forward approach to establishing data communications with the embedded PC. The PLC's on three of the five proposed sites are Modicon Quantum series and the other two are GEC Rx7i PLC's (BEC Engineering. 2008b). All these PLC's can communicate over an Ethernet network using the MODBUS TCP/IP application layer protocol.

For the embedded PC to communicate with PLC equipment, software was required to establish communications using the MODBUS TCP/IP protocol. This was achieved via a commercial software component from 'Automated Solutions' (Automated Solutions 2008). This product is an Active X component which emulates a MODBUS master over a TCP/IP network.

A block diagram of communications protocols is shown below in figure 3.10. While the protocols are MODBUS TCP over Ethernet, the embedded PC acts as the master for communications with the PLC and the slave for communications over SCADA. SCADA communications is discussed in section 4.1.

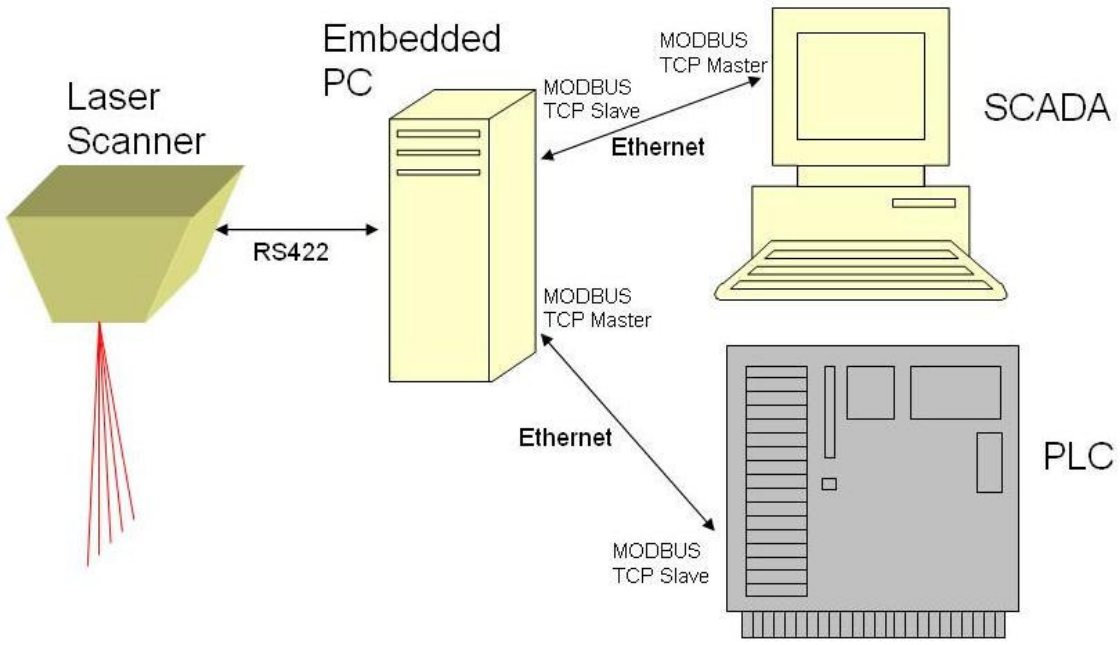


Figure 3.10 – Application Layer Protocols.

3.10 Error Handling

The embedded PC software includes an event log. Any detectable abnormal events, significant events or run time errors are recorded into a text based events log file. Normal events which are recorded include start up and shut down processes. Losses of communications with any of the laser scanner, SCADA, PLC or the database are also recorded. Database anomalies such as missing, corrupt or invalid data are also logged. Despite extensive testing, in the event that a visual basic error does occur each non-critical function has an error handler to allow the program to continue operating. The error handler will record the function where the error occurred and the error message. Appendix viii contains a sample taken from the event log while the software was being tested.

4.0 SCADA

The car dumper is highly automated and controlled directly by using two PLC's. This dumping operation is monitored by an operator in the control room. The control room has a CCTV network to watch the operations. An operator will also monitor the operation via a SCADA/HMI. The existing SCADA system was developed using Citect (BEC Engineering 2008b). The existing Citect system window gives the operator real time access to information from the PLC regarding the unloading operation. Part of this project is to integrate this laser scanning system into the existing site SCADA/HMI system allowing the operator to view the information gained from the laser scanner.

Citect is a fully Integrated HMI (Human Machine Interface) and SCADA (Supervisory control and data acquisition) package. Citect is used to collect data and provide an interface to control specific equipment such as PLC'S or remote terminal units (Citect 2008).

4.1 SCADA Communications

To enable transfer of results and scan data from the embedded PC to the Citect SCADA system a method of data communications had to be established. Citect has the ability to communicate with PLC's and other equipment using a large number of protocols. The existing sites have Ethernet communications networks established and therefore Ethernet was used to integrate this project into the existing infrastructure.

MODBUS TCP/IP was selected as the application layer protocol. The existing sites have PLC's that use the MODBUS protocol. MODBUS is also one of the protocols preferred by the client. MODBUS TCP/IP was selected in order to keep the number of protocols used on site to a minimum. OPC was also considered but rejected by the client. MODBUS is a master / slave protocol and Citect only has the ability to act as the master. Normally a PLC is the slave. Therefore the embedded PC will need to act as the MODBUS slave to communicate with Citect. The relationship between application layer protocols are shown in figure 3.10.

For the embedded PC to communicate using MODBUS TCP/IP protocol as a slave, a commercial software package was also used. A number of suitable software components were assessed to find one suitable for this project. The selected component was the MODBUS TCP slave also from 'Automated solutions' (Automated Solutions. 2008). This product is of industrial quality and is included into the embedded PC software as an Active X object. The object emulates a MODBUS slave device such as a PLC or RTU. The object contains MODBUS standard register types such as input registers, holding registers, inputs and coils. The Active X object accepts MODBUS standard read and write instructions from a MODBUS master device. This product is also the complement to the MODBUS master software used to communicate with PLC's discussed in section 3.9. One development licence was procured for use in this project.

Each variable used in Citect needs to be defined, setup and configured. Citect documentation refers to variables as tags. Each tag needs to be configured in a database so that citect knows how and where to retrieve the information. Normally when communicating with a PLC, a tag would represent a system variable such as a process variable, manipulated variable etc. Tags can be almost any of the basic types such as integers, real numbers, booleans or strings. Each tag is defined in terms of its name, variable type, IO device, protocol, and MODBUS slave register number. Appendix ix has a full list of tags with their variable type; MODBUS register number and a description of their purpose.

To keep network traffic to a minimum raw polar co-ordinates are sent from the embedded PC rather than the Cartesian co-ordinates which occupies more bytes. These co-ordinates are converted in to Cartesian values in Citect before being sent to the charting package. In the polar array each co-ordinate measured is separated by half of a degree, so the polar angle is implied by the position of the co-ordinate in the polar array. This means that polar co-ordinates only require one array of values where as Cartesian co-ordinates required two arrays for X and Y values. The polar co-ordinates are also of type integer whereas Cartesian co-ordinates are of type double. Overall a single Cartesian co-ordinate will require 8 bytes of data and polar co-ordinates will only require 2 bytes. Conversion to Cartesian co-ordinates is calculated by Citect for use in the Citect GUI in addition to the conversion done in the embedded PC for the calculations.

4.2 Charting

The main feature of the graphical user interface is the three dimensional chart of the cars. This chart will show a profile of the cars on the X and Y axis while the third (Z) axis will be time. The plots will be taken while the ore car is moving so the Z axis will resemble the third dimension of the ore cars.

The chart has been developed using the graphing package proessentials V6 from ‘Gigasoft’ (Gigasoft 2008). A number of commercial charting packages were assessed to find the most suitable one for incorporation into this system. Proessentials was developed for use by the science and engineering industry (Gigasoft 2008). Proessentials has a number of different chart and graph types to select from(Gigasoft 2008). Implementation into the system software is via an Active X object(Gigasoft 2008). It was selected because of its ability to do three dimensional charting with real time updates. One development licence was procured as part of the normal process for equipment procurement.

Part of the package from ‘Gigasoft’ includes a number of code samples in several languages including several C variants and visual basic 6. This project uses the 3D real time charting component of proessentials and is programmed using the visual basic development kit within Citect. Citect is programmable using its own native language Cicode or a variant of visual basic called Citect VBA. Citect also has the ability to be able to access Windows Active X objects. The charting component of this system was initially developed using examples of source code from Gigasoft (2008) implemented into visual basic 6 because there were no code examples for citect VBA or Cicode. Once a working algorithm had been developed the source code was adapted into Citect VBA.

The Citect SCADA in this project has three Citect VBA functions developed for controlling the graphing package. The first one to initialize the chart, the second function is to add new chart data whilst the third will refresh the chart for a new ore car. There is also a main function from where the charting functions are called. The three function are now discussed in detail.

The initialize() function is called when the window is opened by the user. This function sets all the necessary variables and invokes all Active X methods required to draw the chart on the window. A number of plot styles are available on the package. In this project a polygon plot style is used which inserts a coloured polygon between each plot point. The polygon colour is determined by the Y value, therefore the chart is presented as series of contours. A minimum value has been set to 500 millimetres preventing any values below 500 from being displayed on the chart. Thus the ground profile is prevented from being displayed. All scales are set and fixed. The axis markers and legend have also been turned off. The number of X and Z points are also established in this function. The number of X points is twice and scanning window angle as discussed in section 3.2. because the points are 0.5 degrees apart. The number varies between sites and is gained from an input argument to the function. Currently the number of Z points has been fixed at 50. The plot values are stored using a 2D matrix for each of the X,Y and Z axes . All matrixes are 50 by twice the scanning window. All of the initial Y values are set to zero which will result in a blank screen.

The update() function is called whenever a new data flag is set by the embedded PC in one of the MODBUS slave registers. The Citect system communicates with the embedded PC using MODBUS TCP protocol. Citect only has the ability to act as the master so there is no way to generate a new data event from the embedded PC. Citect scans the flag at regular time intervals to check for new data. This poll rate can be slower than the rate that data is coming from the laser scanner. Therefore it becomes necessary to have a buffer which can store several sets of scan results. The size of the buffer is set in one of the MODBUS slave registers by the embedded PC . The embedded PC will increment this value if the previous scan results have not been collected by Citect. Citect acknowledges collection of new data by resetting the new data flag. The update() function accepts arrays of new X and Y values. Arrays are loaded into the chart using one of the chart methods which automatically shuffles all the previous values along one point in the Z plane. The oldest values are deleted and the latest set added. This function also redraws the screen.

The refresh() function replaces all the y values with zero before redrawing the screen. This gives a blank screen when a new ore car is under the scanner. This function is called whenever a new car flag is set by the embedded PC on the MODBUS slave register. This flag is reset within Citect.

Function NextVal () is invoked by the Citect event generator every second whenever the Car Scanner page is open. This function checks the new data flag for new data and then proceeds to unload the MODBUS slave registers containing the arrays of polar data. Once the array of polar co-ordinates are converted to Cartesian co-ordinates these values are passed to the update () function.

The Citect SCADA screen also displays the results of the last car that was scanned including the quantity of residual ore, imbalance of ore, the level of the car and degree of car misalignment. These values are displayed directly from the appropriate tags. Alarm set points for these values can be entered onto the Citect HMI screen by an operator. The embedded PC contains the code to determine if any of the values exceeded its alarm set point. Should a set point be exceeded then the appropriate alarm flag will be set by the embedded PC. Citect will generate an alarm event when one of the alarm flag registers is set. Set points values are inserted directly from or to the corresponding Citect tag. The density value used to convert the volume of ore to a mass value can also be changed by the operator. Density is treated as a set point and is also inserted directly into the Citect tag. The ordinal number of the present car extracted from the PLC is also displayed on the Citect HMI as well as the car class type. The Citect HMI screen can be seen in figure 4.1.

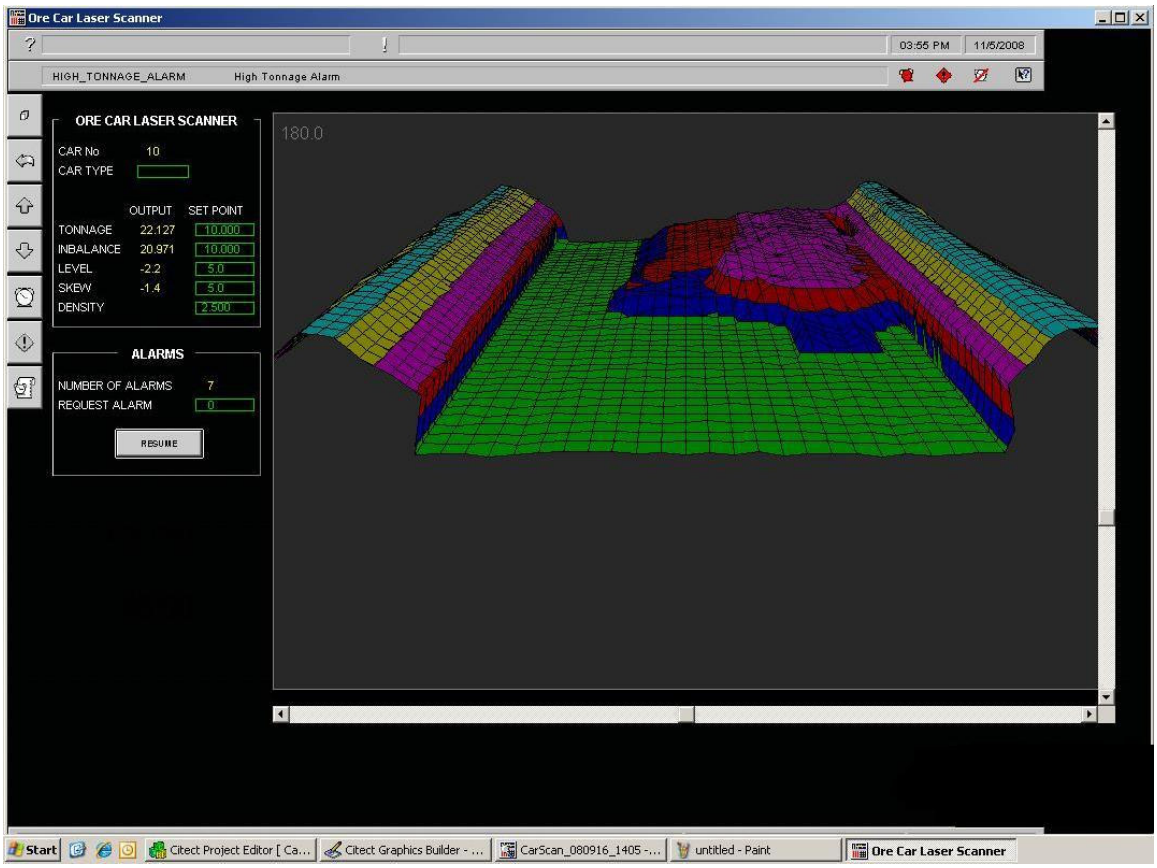


Figure 4.1 – Citect SCADA/HMI screen.

4.3 Buffering scan data

Event timing was by far the most difficult problem associated with data communications over the SCADA network for transmission of charting data. Citect is essentially a GUI package to interface to control systems for the process control industry. Citect was not really designed to load and store data. One of the challenges during this project was to find a way of buffering the data from the laser scanner to be displayed on the Citect GUI. MODBUS TCP/IP is a master slave protocol and Citect only has the ability to act as a MODBUS master. This prevents the laser system from generating an event in Citect which can be used to trigger charting functions. For Citect to determine if there is new data, Citect must poll the embedded PC at regular intervals and download any available data. Because the poll time is slower than the scan rate, provision for downloading several scans at once must be catered for.

Several buffering algorithms had to be trialed to get the transmission of this data to an acceptable level of operation. The first algorithm that was tested monitored the new data flag and created an event when this flag was set. This algorithm used the Citect event generator. Sometimes this event generator failed to detect a change in the flag status. The code relied on the event to reset the flag therefore when an event was not detected the system was left hanging thus preventing any data transmission from occurring. The cause of this problem was not investigated but worked around by polling the flag.

The second algorithm tested polled the flag at one second intervals. While this method reliably generated the event, it often lost an array of polar co-ordinates. Investigation of this problem revealed how Citect updates its variable data. The Citect variable tags are updated at intervals determined by an algorithm hard coded in the Citect system. Therefore, when the value of a variable is requested in the visual basic code the value is taken from the local RAM and not over the network from the PLC or embedded PC. The consequence of this is that the value could be out of date.

Citect has its own algorithm to find the most efficient method of retrieving register values from a device. This algorithm is referred to as smart blocking in Citect Pty Ltd (2003). The intention of smart blocking is to minimize network traffic only refreshing tag values as they are needed and retrieving them in blocks rather than as individual variables. The idea came about because Ethernet packets contain a large quantity of overhead data used for addressing, control, checksums etc. Citect will block tags where the tags are on the same I/O device, have the same read request instruction and where the registers are within 120 words (240 bytes)(Citect Pty Ltd, 2003). The 240 byte limit is due to the maximum size of a MODBUS Ethernet packet. The MODBUS driver inside Citect has its own cache which contains a queue for incoming data requests. After a specified time the driver will use smart blocking to group the requests made. Then the driver will request multiple data values from the I/O device in one instruction (Citect Pty Ltd, 2003). A MODBUS request contains an instruction (eg read), the first register affected and the number of consecutive registers to be affected (Citect Pty Ltd, 2003).

In order to understand how this algorithm operated, code was written for the embedded PC which used the request from Citect to generate an event. This event recorded the registers requested by Citect, the time and the value of the buffer size at that time. These values were saved into a text file for examination, appendix x is a sample of this data. It was determined that between the time the buffer size was read by Citect and the embedded PC received the acknowledgement from Citect, data was added into the buffer. Any data placed into the buffers during that time was lost and this explains the missing arrays of values.

The third algorithm tested made use of the 'Automated Solutions' MODBUS Slave event generator to determine which buffers had been collected by Citect. Any buffers not collected were moved to the front of the embedded PC queue for collection on the next cycle. This algorithm resulted in the frames getting out of order. Investigation of the request file showed that if a new frame needs to be inserted into the buffer while the previous set of frames are still being read then the new frame could end up either side of any uncollected frame. This depended on which order the operating system decided to process the instructions.

The fourth algorithm tested also made use of the MODBUS request event generator to update the registers values. The event generator flushed the buffer contents to make room for new data. This algorithm resulted in duplication of frames. Investigations of the event files showed that Citect does not always request new data from the MODBUS slave when new data is requested by the Citect programming. Sometimes the old values are reused resulting in the duplicated frame.

The fifth algorithm that was tested did not rely on the buffer size register. A fifth algorithm used indexing to find updated scan data. An index value representing the number of times the current car had been scanned has also been included with the scan data. This value was placed in the array of polar values at position zero inside the block of 120 to use the Citect smart blocking to our advantage. This ensured that the Citect collected this value at the same time as the remainder of the polar array. The Citect algorithm was modified to collect all of the buffers but only to display the ones with a valid index number and discard any frames which had already been displayed on the GUI. To prove that the new algorithm worked, the index value was temporarily displayed on the GUI along with the index value used to track the Z axis. This was used to monitor the number of lost scans. This algorithm is far better than the others, as it did not duplicate any frames, inserted frames into the correct order, but still dropped approximately two percent of frames.

The sixth algorithm was an amended version of the previous algorithm. This algorithm called the tag values directly from the embedded PC instead of using the values from the local RAM by calling the Cicode functions tagwrite() and tagread(). This algorithm lost approximately one scan in every 200 scans. The loss generally occurred at the beginning or the end of a new car when all the registers and the GUI are being refreshed. For this application 1 in 200 is an acceptable loss in comparison to the first algorithm and better than the first algorithm which lost approximately 25 percent of the scans.

4.4 Viewing historical results

Included in the scope of works for this project is the requirement to view previous alarms. A Citect SCADA operator must be able to read historical information from the database. In the event that an alarm event is generated the results and chart will be saved to the database for access by the operator. This is necessary because the operator may be watching other screens beside the graphing screen. The SCADA screen displays the number of cars which have tripped alarms for the train which is currently being unloaded. At the start of the next train this value is reset to zero. The operator can view any car which triggers an alarm by entering the alarm number into the field and pressing the enter key. This event will set the alarm view flag in the MODBUS slave registers and the alarm number entered onto the appropriate MODBUS slave register . The flag will tell the embedded PC to go into historical mode which will search the database on the embedded PC for both the results and chart plot data of the given alarm number. These values will then be placed on to the same tags used for the live scanning process which is then displayed on the SCADA screen. The operator can then able to see the image of the car and the tonnage, imbalance, car number, alignment and level. A discussion of the algorithm to search for the data has been previously explained in the section 3.7.1. The operator can view live scanning by clicking on the *resume* button or by entering *zero* in the view alarm box. Any invalid entry into the request box will be ignored.

5.0 Project Management

The project fell significantly behind schedule due to several hold-ups relating to the on-site installation of the equipment. These delays meant that the majority of the development was done using the cardboard models instead of the real installation. At the request of the client, the proof of concept testing is to be carried out in a simulated environment using real cars instead of the actual car dumpers. This project still needs to be completed for the final installations but the concept could be applied to other applications beside railways.

5.1 Progress

At the time this report was compiled the system had been developed to a point where it could be tested on site. Initial project planning was to install the equipment on five sites and develop the software as each was being installed. The original project plan failed to materialize due to several hold-ups that could not be foreseen. The first and most significant was a health and safety problem on the site which was nominated for the installation of the prototype. On this site an ablution block had a plumbing problem which caused the effluent to run down an embankment into the electrical cable pits. Contamination of the pits created a hazard for the staff contracted to install the equipment. Therefore installation was delayed until the plumbing problem could be rectified.

There was also a delay concerning the location of the laser scanner. The laser scanner had to be mounted in a position over the cars as they are ejected from the dumper. The dumping operation occurs continuously throughout the day and cannot be stopped without economic loss. The installation of the scanner had to be performed using an elevated work platform in order to provide a safe working environment to the installation contractors. Time taken to organize this equipment resulted in further delays to the project.

A further subsequent delay was at the request of the client. As the installation progressed and an increasing number of client management became aware of the project, the client requested a hold on the project in order for the specifications to be reviewed again.

The installation delays had an impact on the timing of the detailed design work. Some of the staff who was initially allocated to this project had dispersed to other projects. Installation delays also meant that testing had to be performed using the cardboard models instead of on site testing. The result was that the software was at a reasonably completed stage of development before the system could be tested on site.

At the time this report was written, installation of the equipment on the prototype site had been completed but testing and commissioning had not yet begun. The software had been developed using the cardboard models and software testing had progressed significantly. The software was relatively stable and met the functional specifications but still had some minor bugs.

5.2 Scope of work changes

During the development of the laser scanning system, the ability of the laser scanner started to become apparent. At the request of the client, testing would be moved from the workshop in Perth to the clients workshop located near the installation site. The relocation of the testing would enable testing to be carried out using real ore cars rather than cardboard ones. In this way the system could be effectively tested on real ore cars without disruption to the working car dumper. Figure 5.1 shows the proposed location for testing the laser system. The laser scanner will be mounted from the hand rail on the walkway above the canopy. A single car can be pushed by hand through and under the scanner to simulate the car being ejected from the dumper. A description of the tests to be performed in this workshop is contained in appendix xi.

Figure 5.1 – Proposed Workshop test Site

5.3 Future Work

The immediate goal for this project is to complete all the work which has been approved to proceed by the client. This includes the testing of the laser system at the rail workshop on real ore cars but under a simulated situation. Approval was given for the installation of one car dumper as a prototype and then subsequently at four other sites. The project aim is to develop the system to comply with the documented functional specifications as well as being a system which is safe, reliable and accurate. Although the accuracy of the volume and level measurements are sufficient, the alignment measurement requires further development. The software components for this project also require further testing to ensure compliance with IEEE software standards.

The longer term goal for the installations is to use the information gained from the laser scanners as feedback to the control system. Should the scanner find an ore car which contains excessive residual ore then the control system could reverse the ejection and re-tip the ore car immediately.

Having spent a great deal of time researching the limitations of these laser scanners and establishing reliable communications with the scanners, the company now has a template which can be used to implement the scanner for other uses. These scanners are commonly used for anti collision systems, robotic applications and profiling the contents of conveyor belts (SICK 2006b). Another important part of this project was the establishment of communications between the embedded PC, the PLC and SCADA systems. While these were achieved with the common MODBUS TCP/IP protocol, it still required significant development time to reach a point where it could be used reliably. The BEC Company also now has a benchmark for future projects should there be any requirement to incorporate an embedded PC into a control system that contains standard PLC's and SCADA systems. This could be used as a future template for applications in automation, robotics or advanced process control where standard PLC's do not have the required computational power and an embedded PC could be incorporated into control infrastructure.

6.0 Summary

This report discusses the use of a laser scanner to examine the contents of railway rolling stock after having been emptied. The purpose of the scanning was to determine the amount of ore which may remain in the cars, to compare the amount of ore on one side with the other side and finally to determine the alignment of the car. The laser scanner output measurements are fed into an embedded PC which also takes information from a PLC to calculate numerical quantities for the residual ore. These calculation results are able to be displayed on a SCADA workstation using Citect as a three dimensional chart displaying the profile of the examined car.

The underpinning principal motivation for this project is to enable reduction of train derailments. As residual ore in cars has been the cause of previous derailments. This system will benefit all client stakeholders by primarily having a safer working environment for operations staff and to facilitate a reduction of economic loss.

The laser scanner is a non contact measurement device. It contains a fixed laser and a rotating mirror to measure distances in a radial arc. The time of flight principal to determine the distance based from the return time of a pulse of light. Arrays of polar co-ordinates are outputted from the scanner to the embedded PC which converts these to sets of Cartesian co-ordinates before integrating these to determine volume. The volume of residual ore is the difference between the scanned volume of a given car with an empty car.

Volumes are calculated for either side of an ore car and added to give the total volume of residual ore. The individual volumes are compared to determine the degree of imbalance with a measured car. Both these values are scaled by a density value obtained from the SCADA system before they are displayed on Citect HMI and recorded into a database.

Other measurements taken when using the laser scanner are the degree of level and the alignment of the cars. Level is measured by comparing the heights of the two side walls. Alignment is measured by comparing the horizontal centre of the cars at both ends with the centre of the rails. These measurements are taken to assist in finding cars which may be damaged or derailed.

A significant portion of this project included the establishment of communications between all the hardware components. Communications with the scanner was established with a serial RS422 while communications between the embedded PC and both the PLC and SCADA was made using MODBUS TCP/IP over Ethernet.

A three dimensional charting package has been incorporated into the Citect SCADA/HMI system to provide a visual aid for operation staff.

Results from testing found that the volume could be measured to a reasonable level of accuracy. Highlighted was the problem of finding a reliable search tool to find the sides of the ore cars. The measurement for the heights of the side walls was a noisy measurement but quite reliable when using a moving point average. Alignment measurements taken were extremely noisy. Even with several filter methods tested the measurement could only guarantee to an accuracy equivalent to one degree in the polar plane.

References

Automated Solutions. n.d. *ASMBTCP ActiveX Help r2.7*. Santa Rosa: Automated Solutions Inc.

Automated Solutions. n.d. *Automated Solutions Modbus Slave Active X Control Help r1.6*. Santa Rosa: Automated Solutions Inc.

Automated Solutions. 2008. Software Solutions for HMI and SCADA Developers. Available at: http://www.automatedsolutions.com/products/modbus.asp (accessed 8/9/2008)

Advantech. 2007. UNO-3072 Embedded automation computer user manual. Advantech.

BEC Engineering. 2008a. *Minutes of client meeting*. South Perth.

BEC Engineering. 2008b. *Empty Wagon Detection Scanner*. South Perth.

BEC Engineering. n.d. *Management Structure.* Available at: http://www.bec-engineering.com.au/BECProfile.cfm?mode=displayManagementStructure (Accessed 12/11/2008).

Citect Pty Ltd. 2008. *What is SCADA* http://www.citect.com/index.php?option=com_content&task=view&id=40&Itemid=44 . (Accessed 04/11/08).

Citect Pty Ltd. 2003. Citect HMI/SCADA Help Topics.

Connel, John. 1998. *Beginning Visual Basic 6 Database Programming*. Birmingham: Wrox Press

Drexel. n.d. *Interfacing with the SICK LMS200*. Philadelphia: Drexel Available at http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html (Accessed 08/10/2008).

Eliason, Alan and Ryan Malarkey. 1999. *Visual Basic 6 Environment, Programming and Applications*. Indianapolis: Que Education and Training

Gigasoft. 2008. *Net Chart Control Reference*. Available at: http://www.gigasoft.com/netchart/controlcomponentslibrary.htm (Accessed 11th August 2008)

Ion Tecnnology. n.d. *Modicon Modbus*. Available at http://www.eliteas.com/pml/PMLACFILES/Modbus_6200.pdf (Accessed 28/10/2008).

Microsoft. 2008. Visual Basic MS Comm Control. Available at: http://msdn.microsoft.com/en-us/library/aa259393.aspx (Accessed 3rd October 2008)

Odgen, Frank & Fred Cohn. *Moving up to industrial Ethernet : MODBUS TCP* The Industrial Ethernet book Iss 45. Available at http://ethernet.industrial-networking.com/articles/articledisplay.asp?id=2040 (Accessed 28/10/2008).

Petroutsos, Evangelos. 1998. *Mastering Visual Basic 6*. Alameda: Sybex

Rodgers, Stephen. 2007. *Automated Ship Loader*. Perth: UWA Thesis

SICK AG. 2002. *Quick Manual for LMS Communications setup*. Waldkirch: SICK AG. Available at http://www.pages.drexel.edu/~kws23/tutorials/sick/sick.html (Accessed 08/10/2008).

SICK AG. 2006a. *Telegrams for configuring and operating the LMS2xx Laser Measurement Systems*. Waldkirch: SICK AG.

SICK AG. 2006b. *LMS200/211/221/291 Laser Measurement Systems – Technical Description*. Waldkirch: SICK AG.

SICK AG. 2006c. *LMSIBS Configuration software for LMS2xx/LMI400 Version 4.1*. Waldkirch: SICK AG.

Lammert Bies. 2008. *RS422 Serial Information* Available at http://www.lammertbies.nl/comm/info/RS-422.html (Accessed 07/12/2008)

Appendixes

Appendix i – Notes to the accompanying CD ROM.

The accompanying CD ROM contains the source code for the embedded PC, Citect and the raw test data. The embedded PC software was written using Visual Basic 6. In order to correctly run this program a visual basic development kit will be required. The software also uses three Active X components to run which need to be installed on the PC as discussed in this report. Trial versions of these components can be downloaded from the *Gigasoft* website (Gigasoft 2008) and the *Automated solutions* website (Automated Solutions 2008).

The citect source code and backup files are also included. This will require Citect version 5.42 in order to run correctly. The source code for both the visual basic and citect has also been saved in Microsoft word format for viewing only. The test results are saved in both text files and excel format.

Appendix ii – Procedure for changing scanner baud rate

1.Open LMS/LMI user software.

2. From sensor menu select interface.

3. Set the current baud rate (9600 is the default for a new sensor) and comm. port number.

4. from the file menu select receive. If the connection is successful the software will read the scanner's current configuration.

5. From the LMS menu select user group or press F8. In the popup select authorized customer and enter the password "SICK_LMS" . Click on login.

6. From the measurement technique menu select SICK diagnosis.

7. Select OK at the device prompt.

8. On the menu under change operating mode select installation mode and click add. The installation mode should appear to the right .

9. In the menu, select the desired baud rate and click add.

10. In the menu under Definition perm. Baud rate / variant, select Constant baud rate and click add.

11. Click on run to execute the change. Examine the bottom screen for any errors and confirmation of the change.

12. Close the popup and the LMS/LMI user software

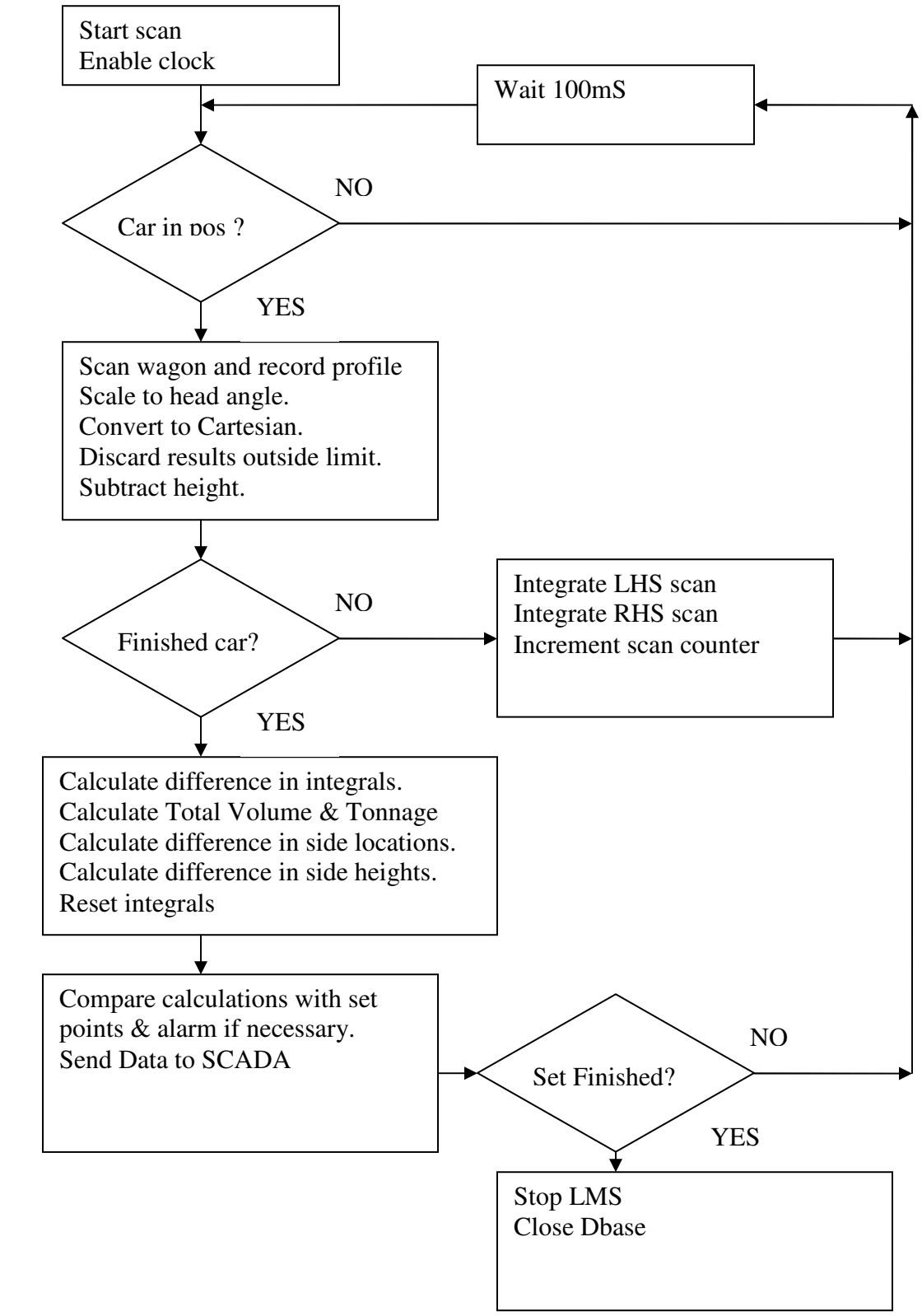
Appendix iii – Sample of raw test data for finding the side

Shown below is a sample of raw data obtained from the program used to determine the most reliable method of finding a point on the side wall which could be accurately be reproduced. Here the algorithm method is shown with the result. In this sample methods 2 and 4 have the most consistent result.

LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 4 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 4
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 3 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 3
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 3 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 3
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 3 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 3
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 4
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 3 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 3
LHside1 (dy) 6 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 9 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 9
LHside1 (dy) 7 LHside2 (y) 7 LHside3 (-dy) 5 LHside4 (dy/dx) 7 LHside5 (-dy/dx) 5

Appendix iv – embedded PC calculation flow chart

The flow chart shows the logical flow for the calculations made by the embedded PC as described in section 3



Appendix v – sample of raw test data, finding the side(2)

These tables are part of the raw data for a program used to compare different ways of finding the side of an ore car. Algorithm 1 finds the mean, algorithm 2 uses linear extrapolation while algorithm 3 uses a moving point average. The objective is to find which method produced the most consistent result.

algorithm 2 - extrapolation method

y1	set point	y2	x1	x1	x result
6.49505392	580	608.130742	-434.149006	-302.448828	-308.606747
31.1973727	580	627.136877	-409.377016	-284.39104	-294.277024
20.4227258	580	635.966419	-411.373977	-282.834154	-294.521243
13.5661323	580	627.136877	-412.64477	-284.39104	-294.243987
29.238346	580	626.155816	-409.7401	-284.564027	-294.243093
17.4841857	580	625.174756	-411.918603	-284.737014	-294.191492
10.6275922	580	625.174756	-413.189396	-284.737014	-294.179422
0.83245857	580	614.383093	-415.004815	-286.639874	-293.833386
54.1976649	580	623.73401	-385.415594	-270.469935	-279.296474
20.8416149	580	632.576803	-391.29716	-268.99016	-279.50208
27.709037	580	632.576803	-390.08625	-268.99016	-279.516172
28.6900973	580	631.594271	-389.913262	-269.15458	-279.488653
28.6900973	580	621.768945	-389.913262	-270.798774	-279.187686
11.031012	580	622.751478	-393.027033	-270.634354	-279.188045
6.12571052	580	608.01349	-393.891969	-273.100646	-278.722601
5.14465022	580	610.961088	-394.064956	-272.607387	-278.814645
385.961142	580	633.242448	-310.259433	-254.485177	-266.494002
28.3193055	580	632.258518	-370.108099	-254.641016	-264.632317
8.66865521	580	620.451359	-373.396488	-256.511085	-264.239602
10.6337202	580	620.451359	-373.067649	-256.511085	-264.242694
10.6337202	580	619.46743	-373.067649	-256.666924	-264.21256

Algorithm 1 - mean X value

x result	y1	y2	y3	y4
-279.403	600.7728	621.7689	625.1748	608.1307
-265.416	581.8202	602.7406	625.6991	627.1369
-264.429	592.658	608.6442	624.7165	635.9664
-265.076	585.7612	608.6442	624.7165	627.1369
-265.368	589.7023	601.7567	621.7689	626.1558
-271.583	609.6281	615.8738	625.1748	
-265.02	588.717	612.5799	621.7689	625.1748
-265.065	594.6285	610.6121	628.6467	614.3831
-251.159	581.6938	605.4663	618.4835	623.734
-256.401	610.3926	619.4674	632.5768	
-256.297	610.3926	621.4353	632.5768	
-250.297	591.5588	609.4073	619.4674	631.5943
-257.151	607.4368	619.4674	621.7689	
-250.554	590.5723	613.3483	619.4674	622.7515
-251.004	588.5993	612.3631	626.3549	608.0135
-250.13	599.4508	621.2303	627.3389	610.9611
-242.149	606.3563	618.2746	633.2424	
-235.862	588.6122	609.3158	619.2598	632.2585
-235.504	600.4643	612.2753	628.1271	620.4514
-235.929	588.6122	611.2888	628.1271	620.4514
-235.766	597.5012	619.1808	618.2746	619.4674
-235.876	600.4643	617.2078	628.1271	606.6763
-235.844	601.4519	617.2078	628.1271	606.6763
-227.872	603.4273	626.0863	625.1713	

Algorithm 3 - moving point avaerage

x result	y1	y2	y3
-302.449	284.737	270.799	285.995
-284.391	270.141	259.316	271.283
-282.834	270.306	258.381	270.507
-284.391	270.306	258.381	271.026
-284.564	270.799	259.472	271.612
-284.737	271.785	258.225	271.583
-284.737	270.799	257.758	271.098
-286.64	269.648	258.069	271.452
-270.47	256.823	244.282	257.192
-268.99	256.667	243.546	256.401
-268.99	256.355	243.546	256.297
-269.155	256.667	243.694	256.505
-270.799	256.667	243.988	257.151
-270.634	256.667	243.105	256.802
-273.101	255.576	243.252	257.309
-272.607	-255.42	241.927	256.651
-254.485	242.368	229.593	242.149
-254.641	242.221	229.178	242.013
-256.511	240.896	228.762	242.056
-256.511	240.896	-228.9	242.102
-256.667	242.368	227.791	242.275
-258.693	240.896	228.068	242.552
-258.693	240.896	228.068	242.552
-241.338	226.821	215.459	227.872

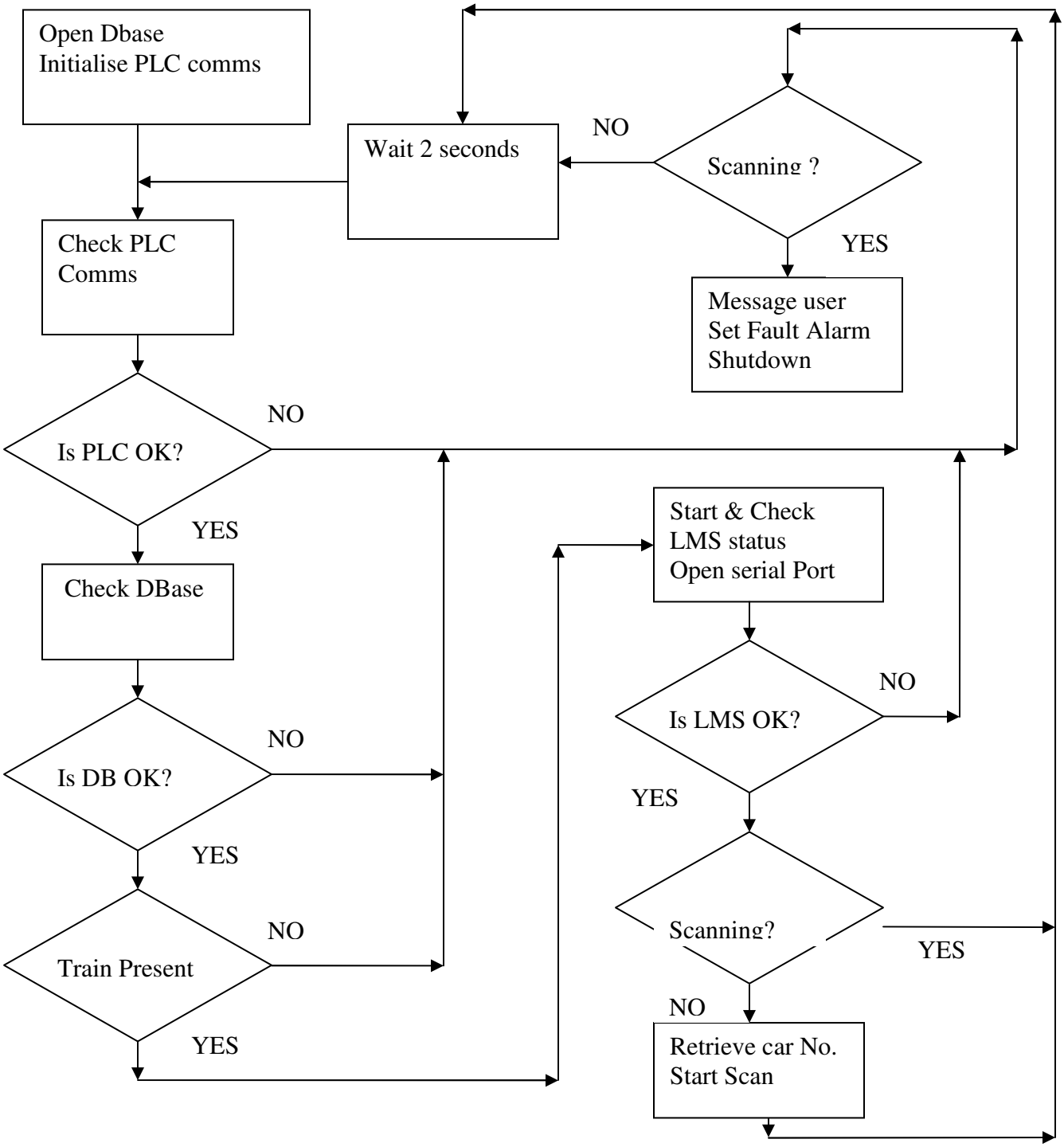
Appendix vi – Sample of database results table

This chart displays a sample of data taken from the results table of the database after testing. The database has been constructed using Microsoft Access and filled using the ADO Active X component. Database components have been discussed in detail throughout section 3.7.

date	time	car_tag	car_type	car_tonnage	car_inbalance	car_LHS_volume	car_RHS_volume	car_skew	car_level
9/26/2008	1:42:33 PM	0	A1	20.8062194198395	8.69415933210045	7.37509468798498	3.02801502193475	-2.30353307115643	1.33078880095954
9/26/2008	1:42:52 PM	0	A1	20.1486551328512	7.47959383850443	6.90706224283891	3.16726532358670	0.797732169177203	0.51791607806968 9
9/26/2008	1:43:08 PM	0	A1	22.7457920254344	12.7173142825196	8.86577657698839	2.50711943572860	-2.40416922619193E-02	1.98244961953208
9/26/2008	1:43:31 PM	0	A1	10.8813750006215	1.32260529718106	2.38969242586012	3.05099507445065	-1.17864609615621	1.46442592979058
9/26/2008	1:46:43 PM	0	A1	25.0464542329022	10.4876368307453	8.88352276591187	3.63970435053921	-2.37916453229628E-02	2.04032700015729
9/26/2008	1:47:34 PM	0	A1	12.6099284235714	0.15196844064659 8	3.1144899957312	3.19047421605450	0.752881994932076	1.99210933745638
9/26/2008	1:48:19 PM	0	A1	14.7645990516344	2.23673111461950	4.25033254156337	3.13196698425362	0.515634887142119	1.15193128974497
9/26/2008	1:54:49 PM	0	A1	19.3481345924744	9.14197428465911	7.12252721928327	2.55154007695371	-1.90985444309607E-02	-4.81330670906743
9/26/2008	1:55:53 PM	0	A1	24.5008915018069	10.3325919529958	8.70837086370068	3.54207488720279	1.42596009907514E-04	2.09979301209 01
9/26/2008	2:00:37 PM	0	A1	22.7108516066739	8.37434325816457	7.77129871620962	3.58412708712734	1.746622067394 4	1.81189661214497
9/26/2008	2:02:04 PM	0	A1	-15.3237788826948	4.83955493577562	-5.04083345461759	-2.62105598672979	-1.91537955479464	0.76144412946786 1
9/26/2008	2:02:40 PM	0	A1	11.9340635448712	2.77741163662733	2.28916297706095	3.67786879537462	1.17555337655384E-02	1.65794689859626
9/26/2008	2:30:29 PM	0	A1	21.2738576426988	8.09029546481101	7.34103827687726	3.29589054447176	0.845440827662326	0.80635483306653 6
9/26/2008	2:32:05 PM	0	A1	17.7008320894628	3.10851282176615	5.20233622780723	3.64807981692415	-1.05505704794551E-02	1.62640984614646

Appendix vii – Alarm flow chart

Shown below is a flow chart displaying the logic behind the system alarm loop. This loop is triggered by a two second timer and checks communications with all parts of the system before turning on the scanner. Any faults are logged into the event text file.



Appendix ix – Citect tag list

Below is a table of all the variables which are passed between the embedded PC and the Citect SCADA system using MODBUS TCP/IP over Ethernet. This table is a cross reference list and is necessary to be able to configure the tag list in Citect so that Citect knows where to get the information from. The MODBUS Register Number refers to the register number of the Active X MODBUS slave object in the embedded PC while the tag name is the variable name used in Citect. Both ends need to know the data type of each variable. This table also contains a comment for the readers reference.

ORE CAR SCANNER TAG LIST

ModBus Registers From	To	Data Type	Citect Tag Name	Comment
1		Boolean	CARSCAN_SCADA_OK	SCADA System status
2		Boolean	CARSCAN_SYSTEM_OK	Scanner System status
3		Boolean	CARSCAN_NEWDATA_CMD	New Scan Data Available flag
4		Boolean	CARSCAN_CARSTART_CMD	Car Start
5		Boolean	CARSCAN_CAREND_CMD	Car End
6		Boolean	CARSCAN_DATAREQUEST_CMD	Request for alarm information
11		Boolean	CARSCAN_TONNAGE_AL	high tonnage alarm
12		boolean	CARSCAN_TONDIFF_AL	high inbalance alarm
13		boolean	CARSCAN_LEVEL_AL	car level alarm
14		boolean	CARSCAN_SKEW_AL	car skew alarm
40001		integer	CARSCAN_LIMIT_V	Scanning window Width
40002		integer	CARSCAN_HIGH_V	Height from ballast to scanner
40003		integer	CARSCAN_ANGLE_V	Scanner head angle
40006		integer	CARSCAN_BUFFER_OP	Size of current Data buffer
40007		integer	CARSCAN_ALARMQTY_OP	No of current alarms
40008		integer	CARSCAN_ALARMREQ_V	No of requested alarm
40011		double	CARSCAN_TONNAGE_OP	Residual Ore Tonnage
40013		double	CARSCAN_TONDIFF_OP	Tonnage Difference
40015		double	CARSCAN_LEVEL_OP	Car Level
40017		double	CARSCAN_SKEW_OP	Car Skew
40021		double	CARSCAN_DENSITY_SP	Density used for calculations
40023		double	CARSCAN_TONNAGE_SP	Residual Ore Tonnage alarm se
40025		double	CARSCAN_TONDIFF_SP	Tonnage Difference alarm set p
40027		double	CARSCAN_LEVEL_SP	Car Level alarm set point
40029		double	CARSCAN_SKEW_SP	Car Skew alarm set point

40101		string	CARSCAN_CARNUM_V	Car Number
40151		string	CARSCAN_CARTYPE_V	Car Type
40101	40220	Integer array	CARSCAN_POLAR1_V()	polar coordinates of scan results
40301	40420	Integer array	CARSCAN_POLAR2_V()	buffer 1
40501	40620	Integer array	CARSCAN_POLAR3_V()	buffer 2
40701	40820	Integer array	CARSCAN_POLAR4_V()	buffer 3
40901	41020	Integer array	CARSCAN_POLAR5_V()	buffer 4
41101	41220	Integer array	CARSCAN_POLAR6_V()	buffer 5

Appendix x – sample of raw data – test Citect communications

This data is a sample taken from a text file which was used to help understand and debug the way in which Citect retrieves data from a slave device using MODBUS. Information from the MODBUS event function was simply saved into the file. Apart from the time, the lines show the size of the buffer at the time, the instruction and the registers affected.

10/23/2008 2:57:31 PM buffer = 0 3 true Read 40151 to 40151
10/23/2008 2:57:32 PM buffer = 1 3 true Read 40011 to 40030
10/23/2008 2:57:32 PM buffer = 1 1 true Read 1 to 16
10/23/2008 2:57:32 PM newdata = true 5 Write 1 to 1
10/23/2008 2:57:32 PM newdata = false 5 Write 3 to 3
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40200 to 40319
10/23/2008 2:57:32 PM buffer = 1 2 false Read 10001 to 10016
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40002 to 40101
10/23/2008 2:57:32 PM buffer = 1 1 false Read 1 to 16
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40151 to 40151
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40400 to 40519
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40600 to 40719
10/23/2008 2:57:32 PM buffer = 1 3 false Read 40800 to 40919
10/23/2008 2:57:32 PM count 52 buffer size 2
10/23/2008 2:57:32 PM buffer = 0 3 true Read 41000 to 41119
10/23/2008 2:57:32 PM buffer = 0 3 true Read 41200 to 41319
10/23/2008 2:57:32 PM buffer = 0 3 true Read 40011 to 40030
10/23/2008 2:57:32 PM buffer = 0 2 true Read 10001 to 10016
10/23/2008 2:57:32 PM buffer = 0 3 true Read 40002 to 40101
10/23/2008 2:57:32 PM buffer = 0 1 true Read 1 to 16
10/23/2008 2:57:32 PM buffer = 0 3 true Read 40151 to 40151
10/23/2008 2:57:33 PM buffer = 1 3 true Read 40011 to 40030
10/23/2008 2:57:33 PM buffer = 1 1 true Read 1 to 16
10/23/2008 2:57:33 PM newdata = true 5 Write 1 to 1
10/23/2008 2:57:33 PM newdata = false 5 Write 3 to 3
10/23/2008 2:57:33 PM buffer = 1 3 false Read 40200 to 40319
10/23/2008 2:57:33 PM buffer = 1 2 false Read 10001 to 10016
10/23/2008 2:57:33 PM buffer = 1 3 false Read 40002 to 40101
10/23/2008 2:57:33 PM buffer = 1 1 false Read 1 to 16
10/23/2008 2:57:33 PM buffer = 1 3 false Read 40151 to 40151
10/23/2008 2:57:33 PM buffer = 1 3 false Read 40400 to 40519
10/23/2008 2:57:33 PM count 54 buffer size 2
10/23/2008 2:57:33 PM buffer = 0 3 true Read 40600 to 40719
10/23/2008 2:57:33 PM buffer = 0 3 true Read 40800 to 40919
10/23/2008 2:57:33 PM buffer = 0 3 true Read 40011 to 40030
10/23/2008 2:57:33 PM buffer = 0 3 true Read 41000 to 41119
10/23/2008 2:57:33 PM buffer = 0 3 true Read 41200 to 41319
10/23/2008 2:57:33 PM buffer = 0 2 true Read 10001 to 10016
10/23/2008 2:57:33 PM buffer = 0 3 true Read 40011 to 40030

Appendix xi – Site acceptance test procedure

A formal part of testing the laser scanner system includes ***Proof of concept*** testing. These steps will describe the technical steps to set up and test the system on the testing site and form part of the formal test document.

Scanner Calibration procedure:

Height
The height of the scanner is to be measured from the ground level to the scanner. Ground level is taken to be the surface level where the finished surface beneath the scanner is concrete. The height of the scanner above the ground is to be measured to the base of the LMS scanner unit using a tape measure. This value is to be entered into the Citect setup page. Confirm this value is correctly read by the embedded PC.

Head angle
The head angle is taken to be the angle in degrees of the scanning plane to vertical. This angle can be measured as the angle of the scanner base away from true horizontal. An alternative method to determine this angle is to adjust the angle parameter on the embedded PC or by physical adjustment of the angle on the scanner unit itself until the ground level as read by the scanner is equal to zero.

Scanning window
The scanning window is the angle in degrees from vertical to the outside of the ore car underneath the scanner. Measured values read by the scanner taken outside this window are discarded by the embedded PC and not displayed on the Citect HMI. This angle is required to be wide enough to include the top of the side wall of the ore car. The value also needs to include a safety margin of 2 degrees to allow for variations between ore cars. In the event that these values are different for each side of the ore car, the greater value shall be used.

Level
A physical level adjustment is provided with the LMS scanner. The required level is relative to the railway iron as distinct from level obtained from gravity. The scanner shall be taken as level when the numerical level measurement on the embedded PC GUI or SCADA is equal to zero. This value shall be accessed as the mean of at least 10 ore cars.

Procedure for profiling empty ore cars :

For each type of ore car which can be emptied on a given installation a database with parameters of each type needs to be developed. The database is located on the profiles table contained in the LMSdata.mdb file which can be opened using microsoft access. Each car type has four fields which requiring entries.

Car type
The car type can be any string value used to commonly identify the class of ore car.

Length
The length of the ore car is to be taken as the internal length of the car. This length shall be determined from either mechanical drawings or measured by physical means and using a suitable measuring instrument. The measurement shall be taken at a point 1100 mm below the top of the car.

Left hand side volume
The value of the left hand side volume is the volume of half an empty ore car as measured by the laser scanner. In the same way that tare is the measured mass of an empty vehicle, a volume is required that represents an empty ore car. The procedure to determine this value is as follows :

1) set the value in the database to zero.
2) Measure the volume of a sample car using the laser scanner. Note this value can be obtained from the GUI on the embedded PC or from the results table of the LMSdata.mdb database.
3) The value to be used shall be the mean of at least 10 test results.
4) Save the mean in the profiles table.

Right hand side volume
As per left hand side volume

Procedure for setting alignment offset.
Alignment offset is the distance between the centre of the railway irons and the centre of the laser scanner as seen from true vertical. This value is necessary to calculate the alignment of the ore cars. The value is stored in the set points table of the LMSdata.mdb database. The procedure to determine this value is as follows:

1) set the value in the database to zero.
2) Measure the alignment of a sample car using the laser scanner. Note this value can be obtained from the GUI on the embedded PC or from the results table of the LMSdata.mdb database.
3) The value to be used shall be the mean of at least 10 test results.
4) Save the mean in the setpoints table.

Testing procedures :

Test for zero tonnage
Confirm that an empty car outputs a tonnage value of approximately zero after being pushed under the scanner

Test for zero imbalance
Confirm that an empty car outputs an imbalance value of approximately zero after being pushed under the scanner

Test for zero level
Confirm that an un-deformed car outputs a level value of approximately zero after being pushed under the scanner

Test for zero alignment
Confirm that an un-deformed car outputs a front alignment value of approximately zero after being pushed under the scanner

Test for zero alignment
Confirm that an un-deformed car outputs a rear alignment value of approximately zero after being pushed under the scanner

Test for tonnage
Confirm that a car containing ore outputs a tonnage value which represents a fair assessment of the amount of ore which can be seen in the car after being pushed under the scanner
.

Test for Imbalance
Confirm that a car containing ore which is predominantly located on one side of the car outputs an imbalance value which represents a fair assessment of the amount of ore which can be seen in the car after being pushed under the scanner
.

Test for level
Using hydraulic bottle jacks or equivalent equipment, one side of the ore car body will be jacked up from the bogie. Chocks will be placed between the bogie and the body so that the side wall is 100mm higher than normal. Confirm that the measurement for level is approximately 100 mm after the car has been pushed under the scanner

Test for front alignment
A length of timber at least 2 meters long and 100mm thick is to be clamped to the top of the side wall on the outside. After the car has been pushed under the scanner confirm that the measured value for alignment is approximately half of the thickness of the wood.

Repeat above for each corner (4 of)